AN INTRODUCTION TO PROMISE THEORY

FOR MAN AND MACHINE

Jan. A. Bergstra and Mark Burgess

DRAFT October 11, 2013

Contents

1
What Does Promising Mean?

This book is about the theory of *promises* — an unusual topic, by any account. For something that so permeates daily life, the subject of promises seems to be curiously hidden away in academic literature. With more than a passing effort, one can find references in areas like philosophy, economics, law, and more casual metaphors for promises used in computer science.

Why should promises be so neglected? One possible reason for this is that the notion of promises is surprisingly difficult to formalize, requiring an even mixture of semantic and dynamic components. A more likely explanation, however, is that the world has been much more focused on the law-giving paradigm of *obligation*. In literature, promises are very often waived completely in favour of obligations. We seem to have a tendency to imagine imposing our will rather than seeking cooperation. Some authors claim that a promise merely implies an obligation to keep the promise, and therefore the notion of a promise is redundant. We shall show that this view is far from the case, and more importantly it is unnecessary. Promises are a fully independent concept, and one that brings great insight to a wide range of phenomena.

1.1 Moral philosophy

In philosophy, the concept of promises has been discussed for the most part in the connection of morality. It comes with the implicit assumption that a promise implies an obligation (called the 'promissory obligation') to 'keep' the promise. There has been a recent revivial of interest in this subject[Ati81, She11b], at the time of writing.

We shall contend that the notion of a promise can be divorced entirely from the subject of morality and obligation, and that there exists a perfectly consistent

and indeed simpler theory that only concerns the management of expectations. An obligation is a sense of foreboding about potential consequences of not acting, and adding such a sense of morality adds compounding layers of subjectivity to an idea that is already bursting with it; moreover, it limits the discussion to promises that refer to future actions[1]. This brings confusion in our view, though philosophers may wish to use our formalized theory as a basis for introducing a moral dimension to the discussion later. The benefit of our somewhat pragmatic approach, we believe, is that it lends itself to a rudimentary mathematical formulation, thus offering a particular kind of clarity usually witheld for engineering disciplines.

The moral discussion of promises is often traced back to the Scottish philospher David Hume[Qui98, Hum78], who saw promises as something of a mystery. If a promise confers an obligation (to keep it) in the view of society then Hume believed this was paradoxical as one cannot oblige oneself to obey. According to our definitions, Hume's apparent paradox is resolved by a discussion of the *scope* of a promise, i.e. the set of individuals who know about a promise (see next chapter).

A privately held intention, such as Hume imagines, certainly carries no sense of foreboding (unless one is prone to self-flagellation), as there is no independently rationalizable penalty to withdrawing it. However, once an intention is made public (with wider scope than the originator of the promise) the intention becomes a promise proper, and the public knowledge may invoke feelings of peer-admonition should the promiser renege on his or her word. By developing this notion of scope, we avoid bringing morality into the discussion – though, this example suggests one way in which the moral discussion might derive from more elementary considerations.

Another problem with the moral dimension to promises is that one cannot transfer the notion of promises to inhuman objects, as one frequently does in normal conversation, e.g. 'the weather promises to be rather inclement this morning', or 'the promise of happiness awaits us'.

The theory of promises espoused in this book avoids many of the conundrums of a moral discussion by requiring only the concept of value, and by replacing moral authority with a more Darwinistic view of economics. Our theory is somewhat practical, as we show how notions of obligation may be constructed from a purely subjective points of view (as voluntary submission), which is curiously related to the authority enhancement hypothesis in philosophy[She11b] (see section 3.4.5). It therefore seems to be of fundamental utility as a toolkit for discussing subjective claims.

In summary, our basic departure from moral philosophy is that promises are

not so much about morality as imposed onto the promiser, but about *information* and the management of expectation seen from the vantage points of both the promisee, and indeed any other individuals who might be privy to the information contained in the promise's content.

1.2 The common or garden promise

In our view, the concept of a promise is not a difficult one – in fact it is a much simpler concept than that of obligations. The tendency to favour obligation over promise probably originates in historical convention, but the lack of independent attention is unfortunate as promises have both interesting theoretical properties and great practical value[2].

The notion of a promise is intuitive but it requires some care. We can talk about promises in their mundane meaning and in more specialized interpretations that may be applied equally to humans or to machines and other inanimate objects. In these pages we present a consistent understanding of promises, and explain why there are both practical and theoretical advantages to their use over obligations. We ask the forbearance of readers in presenting the fundamentals and motivations at some length.

1.3 Where promises fit into the scientific realm

Promise theory must find its place between a number of fields: mathematics, logic, theoretical physics, theory of computation, software engineering and philosophy. Obviously finding that place may take many years and may involve a series of successive authors reworking successive designs of the theory.

We can first specify what promises are not. They are not boolean values, numbers, infinitary quantities, elements of Hilbert spaces, or elements of data types from computing. These things are all different from promises: we may refer to them as so-called data, the existence of which is assumed as a precondition for work on promises. Data are needed to define worlds or spaces in which promises can exist but such words or spaces are not themselves promises. Data can be denote in formalized notations by means of data expressions.

Propositions (and predicates) are the semantic building blocks of logic. These can be used to denote and design a rich variety of assertions about spaces which themselves are set up by means of the data objects just mentioned. Propositional statements are the syntactic appearance of propositions. Many notations for propositional statements exist. Propositions differ from data and more importantly propositions differ from promises.

We will distinguish promises from promise statements. Promise statements are syntactic expressions used to denote promises. Then promises proper embody the meaning of promise statements. We will provide a syntax for promise statements and there will be many examples of its use. There is nothing new about the language of promise statements, but the concept of a promise requires careful attention. We will now indicate why promises may be considered a new class of objects. A promise:

- is made (issued, documented) by an agent called the promiser,
- involves some state to be or to have been achieved or maintained or some activity to have been or to be performed
- is (potentially) perceived by a group of agents, constituting together the scope of the promise,
- is directed towards a further agent, called the promisee.
- can be kept, or not kept (this requires that some time/space span/frame is given),
- broken (not kept at some moment in such a way that whatever happens it can't be kept anymore),
- withdrawn, (by the promiser),
- is assessed (e.g. kept or broken) by a variety of parties including promiser and promisee and members of the scope.

Thus in order to deal with promises in a meaningful way the following ingredients must be at hand:

- agents, (group of agents),
- time/space frames,
- description of behavior, target state, stable state, (specified by means of type and body)
- assessment mechanism

These aspects don't enter the picture when data or propositions are introduced. Thus in comparison with those promises are a new category. We will write in intuitive terms about promises and we will not make an attempt to determine mathematical domains in which promises can be modeled. We assume

that in specific circumstances where the application area is known such domains can be determined by means of well-known methods.

In this book the main task is to develop a notation for promises statements and to consider a range of applications of that notation in order to obtain a rich language for working with promises and a perspective on how promises may be used in practical cases.

1.4 Promises and trust are symbiotic

The usefulness of a promise is intimately connected with our *trust* in the agents making promises. Equivalently we can talk of the *belief* that promises are valid, in whatever meaning we choose to apply. In a world without trust, promises would be completely ineffective.

For some, this aspect of the world promise could be disconcerting. Particularly, to those who are progeny of the so-called 'exact sciences'[3] are taught to describe the world in apparently objective terms, and the notion of something that involves human subjective appraisal feels intuitively wrong. However, the role of promises is to offer a framework for reducing the uncertainty about the outcome of certain events, not to offer guarantees or some insistence on determinism, and in many ways this is like modern theories of the natural world where indeterminism is built in at a fundamental level, without sacrificing the ability to make predictions.

So, at first whiff, promises smell unscientific, but the philosophy of science is clear on only one thing: that uncertainty is the only thing we are certain about, and the subjective role of humans, even with the most discplined will, cannot eliminate the issue of trust or belief from the modelling and interpretation of the world. The task then is to rally what information we have about a scenario and make the best of it. In this regard, promises are allies of probability theory and statistics.

1.5 If promises are so common...

Why are promises so common in the human world? We can only speculate. Promises are about describing the unknown, and one possiblity is that a (rational) agent will always try to maximize its reward and minimize its risk in dealings with the unknown. If promises can reduce the expectation of risk or loss, then they have a function either in limiting fear and stress, or maximizing reward.

Promises, in our interpretation, mix economics, risk, belief and change into

a single concoction. One needs a minimum of economic thinking and philosophical agility to model a 'possible world' described by a promise. Could it be that there has not yet existed a convenient framework whose destiny was to unify these apparently immiscible phenomena? Not so. There are, in fact, many frameworks for describing aspects of the problem that do a reasonable job of elucidating the details.

What has been missing from all of them however is an attention to the autonomy of agents. We make this the starting point from which all else follows[4]. This offers the simplicity of an *atomic theory*, i.e. with indivisible primitives that can be combined to reconstruct more familiar structures.

1.6 How to use this book

This book began as an effort by one of us, in 2004, to understand certainty in computer systems[Bur05]. Since then, both of us have recognized a broader significance to the topic of promises. As the years have passed, we have extended and developed the notions, and our work has caught on by word of mouth, especially in the world of information technology, under the moniker of 'Promise Theory'. Thus, while we are aware of the limits and bounds of our particular theory of promises, it is simpler to keep to this practical name, without offering any promise of its completeness or uniqueness.

What we aim to describe in this volume is some practical ways to use the promise theory we have developed. Indeed, we feel that it has led to many insights, following a clear and simple route, that were not obvious or immediate in other frameworks.

The most important theme in promise theory is the complete subjectivity of experience of every agent in the world. Our Promise Theory is a theory of relativity.

The basic outline of the book is the following:

- We begin by discussing the philosophy of promises to establish some basic concepts.
- Then we define a specific 'idealized approximation' to the notion of a promise that can be described mathematically. This model is simple enough to be general and formal enough to have predictive power.
- We state the essential principles that separate promise theory from other ideas about cooperation.

- We discuss how promises may be assessed, i.e. of whether promises have been kept.
- We then explain the relationship between promises, behaviour, economics and certainty.
- Discuss the symmetries or equivalences of promises.
- Discuss the causation conundrum, and other issues such as 'entangled promises', or complex cooperation with non-local information.

For the sake of brevity, let us start with 'the conclusion'. To apply promise theory to a new problem, one starts like this:

1. *Identify the key agents of intent*.

   An agent is any part of a system that can change independently. To get this part of the modelling right, we need to be careful not to confuse intentions with actions or messages. We are not talking about the moving parts of a clock, or even an Internet search request, but the motivations that play a role in making everything happen. Actions may or may not be necessary to fulfil intentions. Maybe inaction is necessary!

   To be independent, an agent only needs to think differently or have a different perspective, access to different information, etc. This is about the separation of concerns. If we want agents that reason differently to work together, they need to promise to behave in a mutually beneficial way. These agents can be humans (as in the business-IT bridge) or computers (as in multi-tier server queues).

2. *Deal with the Uncertainty*.

   There is no absolute certainty. Promises might or might not be kept, so plan for that. Even a machine can break down and fail to keep a promise, so we need to model this. Each promise will have a probability associated with it, based on our trust or belief in its future behaviour.

3. *Turn 'requirements' into autonomous promises*.

   The goal in cooperation is to ensure that agents make all the promises necessary so that some imaginary on-looker, with access to all the information, would be able to say that an entire cooperative operation could be seen as if it were a single entity making a single service-promise.

   How we coax the agents to make promises depends on what kinds of agents they are. If they are human, economic incentives are usually the

answer. If the agents are programmable, then they need to be programmed to try to keep the promises. We call this voluntary cooperation.

Is this crazy? Why not just force everyone to comply, like clockwork? Because that makes no sense. A computer follows instructions because it was constructed voluntarily to do so. If we change that promise by pulling out its input wire, it no longer does. Cooperation is voluntary in the sense that it cannot be forced by an external agent, without perhaps actually attacking the system to compromise its independence.

4. *Eliminate conflicts of intent.*

   If all agents share the same intentions, there would be no need for promises. Everyone would get along and sing in perfect harmony, working towards a common purpose. The fact that the initial state of a system has unknown intentions means we have to set up things like 'agreements', where agents promise to behave in a certain way. This is what we call orchestration with relativity.

We hope to justify this summary in these notes, with a semi-formal account of a particular and pragmatic model of promises. This model has already proven very useful over these past ten years, and we hope that it will continue to be useful to others.

1.7 Notation and conventions

We use the English term *promiser* for the giver of a promise, as opposed to the legal term *promisor* often used in the moral literature of promises.

For mathematical expressions, unless otherwise stated, we shall use the Einstein summation convention, that repeated indices are summed, i.e. $A_{ij}v_j \equiv \sum_j A_{ij}v_j$. We shall sometimes make use of the reduction notation common in mathematical logic to write

$$\frac{\text{Assumptions}}{\text{Conclusion}}. \tag{1.1}$$

We use this notation before we necessarily want to imply algebraic equivalence.

When we indeed want to express an equivalence, we use the terms:

- $\equiv$ identity to
- $\sim$ approximately the same as

- We use the vertical bar | as in conditional probability to mean 'if' or 'given that', so $b|c$, means 'b given that c is true'.

- The symbol $\wedge$ represents 'OR' (think inclusiveness going up).
- The symbol $\vee$ represents 'AND' (think inclusiveness going down).
- We denote a promise body b subject to a conditional expression c by $b|c$.
- We denote the truth of a conditional expression c by $T(c)$.
- We denote the falsity of a conditional expression c by $F(c)$.
- We denote knowledge of a promise by $\Psi(\pi)$.

Part 1
Fundamentals

2
Promises, Obligations and Intentions

Promises prime our expectations about the behaviour of people and things. They tell us what to expect before we have had the time to learn by experience. Promises act as an anchor by which to start a learning relationship between the maker of a promise and those who will verify its truth in the future.

In modern life, promises are often treated quite lightly. Our expectation would be that promises are not always meant to be kept, but are more of a vague kind of thinking aloud about our intentions. This is an unusual idea in a study that aims to be scientific. We expect certainty and reliability, but in fact it is not a problem. First, one could say that science does not promise those qualities either: its chief function is to be clear about uncertainty, not to eliminate it. A theory of promises taking the view that promises were always kept would not be a very good theory.

2.1 Example promises

Below are examples of the kinds of statements we shall refer to as promises. Let us begin with everyday statements and progress gradually to the kinds of abstract promises that we would like to use in a variety of technical scenarios.

- I promise you that I will walk the dog.
- I promise you that I fed your cat while you were away.
- We promise to accept cash payments.
- We promise to accept validated credit cards.
- I'll lock the door when I leave.

- I promise not to lock the door when I leave.
- We'll definitely wash our hands before touching the food.

These examples are quite uncontroversial. They are easily found in every day life, spoken by humans or posted on signs.

It is easy to see that promises have certain characteristics. They are made voluntarily by an individual (the promiser or promisor), to a recipient (the promisee), and they have a 'body' of description that details what the promise is about. We recognize that other parties may also know of the promise between promiser and promisees. These aspects will have to be captured in a model of promises.

2.2 Promises by inanimate agents

We now want to argue that it is useful to extend the notion of promises to allow inanimate objects and other entities to make promises. This is not a very large step, but it is easier to make with some examples. Consider the following promises that might be made in the world of Information Technology:

- The Internet Service Provider promises to deliver broadband Internet at a fixed for a fixed monthly payment.
- The security officer promises that the system will conform to security requirements.
- The support personnel promise to be available by pager 24 hours a day.
- Support staff promises to reply to queries within 24 hours.

Again these are straightforward promises, which could be described further to be more specific. The final promise could also be restated in more abstract terms, transferring the promise to an abstract entity: "the help desk":

- The company help-desk promises to reply to service requests within 24 hours.
- The weather promises to be fine.

This latter example illustrates the way that we transfer the intentions of promises to 'entities' that we consider to be responsible by association. It is a small step from this transference to a more general assignment of promises to individual components in a piece of technology. For example, we can document the properties of the following tools and technologies in the spirit of this argument:

- I am a meat knife and promise to cut more efficiently through meat.
- I am a logic gate and promise to transform a `TRUE` signal into a `FALSE` signal and vice versa.
- I am a variable that promises to represent the value 17 of type integer.
- I am a command line interpreter and promise to accept input and execute commands from the user.
- I am a router and promise to accept packets from a list of authorized IP addresses.
- I am a compliance monitor and promise to verify and automatically repair the state of the system based on this description of system configuration and policy.
- I am a high availability server and I promise you service delivery with 99.9999% availability.

From these example we see that the essence of promises is quite general. Indeed such promises are all around us in everyday life, both in mundane clothing as well as in technical disciplines. Statements about engineering specifications can also profitably be considered as promises, even though we might not ordinarily think of them in this way.

When an electronics engineer looks in a component catalogue and sees 'resistors' for sale promising to have resistance of 500 Ohms to within a tolerance of 5%, we do not argue about who made this promise or whether the resistor is capable of independent thought. The coloured bands on the component are a sufficient expression of this promise, and we accept it by association. By this reasoning, we propose that the concept of a promise should be formulated in way which allows for all of these uses.

The value of this association is that promises are things that we use to form *expectations* of the behaviour of all manner of things. Such expectations contribute to reducing our *uncertainty* about their behaviour, and this can apply as much to technology as to humans. We therefore take it as given that the concept of a promise is a useful one and consider next how one can formalize promises in the simplest and least assuming way.

2.3 Implicit promise versus explicit promise

The broadest and most far reaching usage of the term promise is in phrases like "the promise of solar energy" or "the promise of nano-technology". Such

promises are implicit in the sense that the existence that a specific promiser is not assumed, and the word promise refers to an expectation or speculation of someone's assessment of what is being promised.

In contrast, an explicit promise from an individual comes by a promiser in an appropriate context. We will only focus on explicit promises below and we will assume by default that promises are explicit.

2.4 Reduction of uncertainty by promises

The value of this association is that promises are things that we use to form *expectations* of the behaviour of all manner of things. Such expectations contribute to reducing our *uncertainty* about their behaviour, and this can apply as much to technology as to humans. We therefore take it as given that the concept of a promise is a useful one and consider next how one can formalize promises in the simplest and least assuming way.

Producing a promise may be more effective in reducing uncertainty than putting forward an assertion that is stated with more certainty. Indeed if an expectation about a piece of technology or about an agent is asserted with absolute certainty, or merely with some quantified probability of being valid, the question immediately arises how that knowledge has been obtained, thereby increasing uncertainty rather than reducing it.

Such existential questions do not arise within a community of users for a piece of technology that has been delivered with a collection of promises to its user, who simply reacts on disappointing performance with a reduction of trust in the promiser. Future promises from that same source will be received with less confidence. Conversely, if the piece of equipment outperforms the promised performance that fact may lead to increased trust in the original promiser. Moreover members of the user community may exchange new promises where better performance of the same equipment is promised on the basis of their experiences with its use.

2.5 Literature about promises

Because of their overriding ubiquity, and practical importance, one would like to have an account of promises that captures their key properties and explains related concepts such as commitment, obligation and intention. There is a surprising lack of discussion about the meaning of promises in the literature as far as we can tell. Although the concept or its relatives have been mentioned in such diverse areas from logic, law and philosophy to economics, information science

and computing, there is no agreement on what constitutes the semantic content of the terms or if there is even more than a tacit relationship between promise, commitment, obligation etc. The most attention has been given to the concept of *obligations* especially in the area of deontic logic. We believe on the other hand that the philosophical implications of promises are far wider than is generally assumed and that there is both a need and a practical importance to clarify them once and for all. Indeed, we will show that notwithstanding our lengthy definitions the concept of a promise is simpler than that of an obligation.

2.6 Promises in philosophy

We will survey some views that other authors have put forward on promises, henceforth understood as explicit promises.

Atiyah [Ati81] suggests that any promise leads to an obligation to keep that promise that is motivated by the threat of tit for tat reprisals. Reciprocation is thus coupled to the idea of promises immediately, which seems to hop over fundamental definitions directly to a discussion of the economics of keeping promises. The obligations are to avoid injury and to reciprocate goodwill. It might be discussed whether incentives are the same as obligations. Atiyah points out however that promising something cannot be necessarily used to create obligation at will. Promises might cause obligations but they can also represent obligations that already exist, i.e. to show commitment to an existing obligation to pay the price of something. e.g. I promise to pay the bearer the sum of 1 pound (in gold). This is only an existing admission of moral obligation. Atiyah maintains, plausibly, that the motivation for promising has changed throughout history. When people make promises, their intentions are culturally bound. Thus a Victorian gentleman's conception of a promise might not fit with that of a present-day child who promises to be home in time for dinner.

Cartwright takes Atiyah's view and asks what might be the point of promises if not to generate the assumed obligation [Car84]. Why do people bother to make promises about things to which they are already obliged? His answer includes the idea that it is a face-saving measure: to mitigate their humility, suggesting that an obligation is interpreted as a kind of attack or levy of force? Alternatively, perhaps the obligation to keep one's promises weighs heavier than the original obligation (I promise you my word as a gentleman not to kill you, even though the law says I am forbidden). Referring to Fried [Fri81], Cartwright points out that the economics of contractual tit-for-tat suggested by Atiyah is tied to promises and not to the obligations they might confer.

The idea that promises are an economic driver of contracts or agreements

as bilateral exchanges of promises is continued in the work of Gilbert [Gil93]. Then Carrillo and Dewatripont have argued that promises can best be understood as a market mechanism for reducing the uncertainty in a moral-hazard game [CD]. This work does not seem to have been pursued. Does a promise increase the likelihood of voluntary cooperation? A number of other works mention the concept of promises in the context of game theory also. In these, the concept of a promise is tacitly assumed to be related to the probability of choosing a particular game strategy.

Scanlon [Sca90] meticulously analyses how and to what extent promises give rise to obligations under a variety of combinations of additional assumptions. In his analysis morality plays a important role and it is implicitly assumed that promiser and promisee are human beings capable of moral reflection.

Zhao et al. [ea05c] provide a comprehensive modal logic incorporating beliefs, capabilities and promises. Unfortunately it is difficult from that work to extant a clear intuition of the concept of a promise that the authors had in mind. It seems that this difficulty is in part caused by the formalist approach taken. In Framinan and Leisten [FL10] order promising is displayed as a standard technical term in industrial workflow management, while at the same time that use of the term promise is considered lacking a sufficiently clear definition.

2.7 Promises in distributed computing

More recently, a different motivation for promises was introduced by Burgess in the context of distributed management [Bur05]. Rather than focusing on morals or even economics as the principal motivator, Burgess uses the promise as a measure of 'voluntary cooperation' as a way of circumventing what we consider to be fundamental problems with logics of obligation for determining system behaviour. Voluntary cooperation is seen as a way of simplifying constraints and avoiding many-worlds paradoxes. He pursues the argument further by emphasizing the role of autonomy of the parts, and argues for a 'promise theory' in which every component in a system that can have unique information or independent action should be viewed as axiomatically autonomous [Burb]. Any cooperation or even subordination of the parts that comes about in an organized system must then be understood as the result of 'honouring' purely voluntary promises to do so. Burgess argues that no matter what one believes about the power to oblige (even soldiers can refuse to follow orders), voluntary cooperation can be used as a pragmatic engineering methodology for mapping out the complexity of a control problem in a way that is invariant with respect to centralization or decentralization of systems.

In computer science, particularly the field of Multi-Agent Systems the concept of *commitments* has been used for some time [WS03, Woo02]. It has been suggested that promises and commitments are the same. However, we shall show that this is not the case. More seriously, the sense in which the term commitment is used in such discussions is more stylized than purposely considered and can only benefit from the discussion in this paper.

2.8 A model of the structure of a promise

Consider the following intuitive idea of what a promise might be: *A promise is an announcement of fact or behaviour by a promiser to a promisee, observed by a number of witnesses (referred to as the scope of the promise), whose outcome has yet to be assessed.*

The promiser and promisee are both assumed to be 'agents', i.e. humans or inanimate objects to which we attach identity in the story of promises. This general description fits the examples that we have already given and gives some clues as to the constitution of a promise, but it also opens up a number of questions that need answering. Already we can see that this apparently basic definition rests on a number of assumptions: that we can observe the outcomes of behaviours and that the outcome of a promise is clear at some single moment of time in the future, to be measured and verified by an observer. A full account of this might include a theory of measurement, but we wish to avoid this level of detail as it binds us to too many details that have nothing directly to do with the issue. Let us instead try to understand to essential characteristics for promises and consider what distinguishes a promise from related matters, such as obligations, commitments and other terms.

The model world in which we formulate promises must have the following characteristics.

- There must be agents in order for promises to exist.
- There must be a promiser (or source agent).
- There must be a promisee (or recipient agent) which might be the same as the source.
- There must be a body which describes the nature of the promise.

We might summarize these attributes with a notation as in [Bur05]:

$$promiser \xrightarrow{body} promisee \tag{2.1}$$

- We can leave the body unspecified, but it must consist of a quality (a type, topic or subject for the promise) and a quantifier (which indicates how much of the realm of possibility for that subject is being promised). For example: promise quality: "travel to work", promise quantity "on Monday and Friday each week".

Finally, what is implicit in the above is that a promise requires the transmission of a message, or at least documentation in some kind of physical form, e.g. a speech act, or a written statement, else it cannot be made known to anyone except the promiser. A promise must therefore have documentation that is made intentionally or otherwise.

What then is a promise before we write it down? We shall refer to this as a *possible intention*. An intention is the basic formulation of a course of behaviour, which is made internally by (or on behalf of) an agent. When an intention is made public, it becomes a promise. If an intention is documented or leaked in some way then anyone has a right to assume it is a promise.

We take it as given that there has to be a source for every promise. A promiser does not have to reveal its identity of course, so witnesses to the promise might not know its source e.g. consider the anonymous threat. There is no reason to deny the existence of a source however. The lack of such information about a promiser is simply a defect in the knowledge of the receiving agent, but one would normally prefer to assume a consistent picture of promises and infer the existence of an anonymous promiser. This justifies our postulating the source.

2.9 The building blocks of promising

Stoljar [Sto52] writes that a promise is an announcement of an intention. and he argues that any offer must be a promise as well. Indeed a key characteristic of a promise is that it documents an intention, so let us explore the idea of intentions in more detail. Intentions turn out to be a lowest common denominator for all of the concepts discussed in this paper and thus have a special importance. Unfortunately from a document one cannot conclude to what extent the intention expressed in it is real. That indeterminacy gives rise to an additional complexity of our definitions.

Since promises involve communication we require a notion of the spread of information amongst the agents. We use the term *scope* (as used in computer science) for this. A scope is simply defined to be a set of agents. For example, the scope of a promise would typically refer to the promiser and a list of witnesses to the promise, e.g. those who heard to utterance or those who saw the written document.

2.9.1 Intentions

In the realm of all possible formulations about agent behaviour the concept of an intention stands out as an important foundation.

Definition 1 (Current intention of an agent A) *A current intention of an agent A is description of a possible behaviour, or goal, or objective, or state of affairs, that is contemplated by A with the understanding that it can be and preferably (for A) will be brought to realization.*

Definition 2 (Possible intention for an agent A) *A possible intention for an agent A is a description of a possible behaviour, or goal, or objective, or state of affairs, that may but need not currently (at the time of qualifying the description) be contemplated or preferably brought to realization by A, and which might be in some (possibly different) circumstances a current intention of A.*

Obviously a current intention is also a possible intention. But if an utterance of A announces a possible intention that need not be a current intention, it may only appear to be a current intention.

The components of an intention are as follows: a source agent who formulates the intention, a target agent if the intention is directed at a potential subject, and a body which explains the quality and quantity of the intention. Only the source of an intention knows about the intention, i.e. the scope of an intention is the source only. There are no witnesses.

Now we must be careful: the set of all possible intentions should be distinguished from actual instances of intentions selected by an particular agent. We shall sometimes use the phrase "possible intentions" to mean this full set of abstract entities to emphasize when we wish to signify a general description of behaviour rather than an individual agent's decision.

2.9.2 Commitments

To commit to something is to make a decision in favour of it. The issue of commitments is therefore about the favourization of intentions. Commitment is a personal decision and has nothing to do with physical representation or communication, thus the issue of commitment precedes any discussion of promises. A commitment has a source, a target and a body, i.e. it is an intention. Like an intention it has no physical representation and does not have a non-trivial scope.

Once an intention becomes a commitment we often assume that some point of no return has been passed in the act of committing (deciding) about the particular intention. i.e. adding the intention to a list of commitments. For example, in

a game of chicken in which two cars drive towards each other to see which one will swerve off first, a driver has committed to not swerving when the decision to not back down has been made [sch60]. This might have certain irreversible consequences, but it is difficult to generalize the idea of irreversibility in examples of this kind. What commitment essentially boils down to is the elevation of some intention beyond an arbitrary threshold. In other words, in the universe of intentions there is a subset of these which we may call commitments.

Definition 3 (Commitment) *Commitments are current intentions that we are committed to. We may call them intended intentions, or equivalently real intentions, intentions that we hold, or committed intentions. The commitment of an intention exceeds its merely being current in that it is stable and persists in time until some achievement of the intention will take place or until some overruling considerations invalidate the commitment.*

When passing by a shop one may feel a current intention to buy a nice gadget, only to be relieved of that intention (or rather its currency) after noticing its price. If however, the price is quite good, but the shop is closed at the time of passing along, then a current intention to buy the same item can become activated with the status of a commitment, only to be terminated when the item has been acquired or when unexpected problems turn out to stand in the way.

2.9.3 Expressing intentions

Making intentions known to other agents is the essence (meta-intention) of promising. We will develop some terminology for such acts of expression. An intention is not necessarily announced by the agent holding it to any other agents. Indeed, we may now *define* any intention that is announced to be a promise. Conversely we notice that any promise that has not been announced is merely a possible intention. Some intentions are desirable while others are absolutely undesirable and an agent might never choose them, yet they are possible intentions nevertheless. The fact that such behaviours can be intended is enough for them to qualify as possible intentions.

Intentions must always be thought of as belonging to a specific agent. Those intentions which are actual plans of the agent are called its commitments. Other phrases for a commitment that we may use are: intended intention, or real intention.

Due to the static nature of our account we pay no attention to the process by which an intention might become a commitment or vice versa.

Definition 4 (Intention utterance) *An agent a produces an intention utterance if A produces an expression of a description of a possible intention.*

What matters for our discussion on promises is intention utterances that seem to be real. That leads to the idea of an apparent intention utterance.

Definition 5 (Apparent Intention utterance) *An utterance expressing a possible intention (of a principal agent) with the contextual appearance of an intention. Apparent intentions, may be drawn from the following range:*

Real intention: *(alternatively: commitment, true intention, or intended intention) what is announced corresponds to wha the agent expects that will happen, or that (s)he will do, or what holds or what will hold.*

In other words the apparent intention is real if it is a commitment (and therefore current).

Incidental intention: *(alternatively: non-committing current intention) what is announced corresponds to why the agent expects that will happen, or that (s)he will do, or what holds or what will hold, but only a the time of expression.*

Indifferent intention: *(alternatively: quasi-intention) the issuer has no current intention corresponding to the utterance, and no current conflicting intention either.*

An indifferent intention is currently contemplated as a possible behaviour, goal, objective, or state of affairs, but its bringing about is not preferred, and thus an indifferent intention is not a current intention.

Deceptive intention: *(also: misleading intention) the announcement might seem to be real for an audience in scope but it is a lie from the perspective of the promiser.*

A deceptive intention is incompatible regarding realization with a current intention, though this may be only known to the principal agent.

Invalid intention: *(alternatively: manifest lie) all observers may notice a discrepancy between what is announced and the facts.*

The invalidity of an invalid intention will become clear to agents in scope of that utterance.

The idea of an apparent intention is that at face value it is like an intention from the perspective of an observer but there is a considerable degree of freedom

in connection with a so-called underlying intention, the existence of which we postulate in the following definition.

Definition 6 (Underlying intention (of an apparent intention utterance)) *Given an apparent intention utterance of an agent, there is an underlying intention (which need not be comprised in the same utterance) as well. We will distinguish five cases, corresponding to the case distinction of intention utterances:*

Real intention: *The underlying intention of a real intention is that same intention.*

Incidental intention: *The underlying intention of an incidental intention is that same intention which is known to be coronet but non-committing.*

Indifferent intention: *The underlying intention of an indifferent intention is empty.*

Deceptive intention: *The underlying intention of a deceptive intention differs significantly from the (deceptive) intention.*

Invalid intention: *The underlying intention of an invalid intention differs noticeably (for observing agents) from the (invalid) intention.*

We will assume the agents keep underlying intentions private. Otherwise new levels of complexity emerge as underlying intentions may turn out to split over the same distinction of four cases recursively.

2.9.4 Promises

A promise is the physical publication of an intention within a certain scope. This suggests that there must be some agent to observe the promise and its outcome which in turn requires the expensive notion of a theory of observation so we shall tackle this issue separately (see section 2.14).

Definition 7 (Promise) *A promise is an apparent intention of an agent, (the promiser or promising agent) the utterance of which has been documented within a scope that goes beyond the promiser.*

According to the definition of intention utterances, a promise brings with it an apparent intention and an underlying intention, and five cases can be distinguished for promises: real, incidental, indifferent, deceptive, and invalid.

Promises thus have scope. Formally intentions also have a scope, but the scope of an intention held by an agent is by definition limited to the agent (source) itself. An intention could be leaked deliberately (e.g. to the press, in order to influence someone). This might be a form of leverage, or an attempt to impose an obligation on some party in scope. However, at the instant an intention expands in scope to encompass more agents it becomes a promise. A so-called letter of intent, for example, is a promise rather than merely an intention.

The time aspect of promises presents further challenges. Intentions can become outdated by events. An event which is found to fulfill an intention documents the implicit promise, since one must admit to the intention in a wider scope. Conversely, the documentation for a promise does not have to last for ever; if documentation of a promise disappears completely, it reverts to being an intention. A promise to oneself is in excess of a mere intention because it has been documented.

The distinction between the promisee and the scope of the promise is key to understanding promises. Suppose someone intends to arrange a surprise birthday party for their friend. Initially this is an intention. The intention is written in a diary or mentioned to a third party and it becomes a promise. The target is not in the scope of the promise however, so the promise remains unknown to the jubilant. However, suppose that before telling anyone else, the promiser destroys all evidence of the promise by tearing out the page of the diary, effectively withdrawing knowledge of the promise, then the promise reverts to being just an intention. But as long as knowledge of the promise remains "out there" in the world, it remains a promise that has been made.

Definition 8 (Promissory obligation) *With each promise of an agent A an obligation is connected, the so-called promissory obligation. It is that obligation to which the agent has become obliged by making the promise.*

Promissory obligations are an important tenet of the philosophy of promises, and we do not deny their existence. However, we will oppose to what we call obligationism.

Definition 9 (Obligationism) *With obligationism we denote the viewpoint that (i) promises are characterized by a unique capacity to (auto)generate an obligation (specifically the promissory obligation) for the promising agent, and that (ii) the essence or content of a promise is fully captured by its promissory obligation.*

Definition 10 (Non-obligationism) *With non-obligationism we denote the belief that obligationism is false.*

Definition 11 (Strong non-obligationism) *With strong non-obligationism we denote the belief that obligationism is false and that in addition the concept of promise may be accounted for without making use of the concept of an obligation.*

Preferring non-obligationism over obligationism, and convinced of the relevance of strong non-obligationism, we will propose a strongly non-obligationist conception of promises. Arguments for (strong) non-obligationism have been detailed in [Bur05]. A recent argument an be found in [BdL13] where payments in a peer-to-peer system for financial transfer an store of value are considered autonomous actions to the extent that such payments cannot be obliged by definition. Payments can be promised however, with the corresponding promissory obligation constituting no more than acquiring some certainty that the payment could be effected if the agent wishes to do so. That promissory obligation does not capture the full content of the promise, however.

2.9.5 Potential prominence of promises

Our belief in non-obligationism implies that conceptually separating promises from obligations is a starting point of our work. However, viewed as utterances promises must be separated from other utterances such as informative utterances, questions, and commands.

If an agent A expresses a fact, say F, meant to be informative for an audience, the similarity with A promising F is significant. However, when stating F, agent A will not try to exercise its influence to bring about that F holds true, not even in order to increase the trust that other agents attribute to A. While it is always acceptable that an agent acts in such a way that its promises are kept, the same is definitely false for informative statements.

For instance: an agent telling an employer that “he will not be present tomorrow because of a flu” need not be absent if the flu fails to hit as expected. Such a message implies no single action that increases its plausibility. What may get lost when viewing informative utterances as promises is objectivity. This provides a clear separation, and at the same time it indicates that in systems that do not measure or assess agent objectivity it is unproblematic to treat each informative utterance as a promise.

When agent A asks question q to another agent, say B in the scope of an audience, then the corresponding utterance need not be considered a promise. Indeed for understanding the notion of a question one has no need of promises. In other words as concepts promises are not prior to questions. However, having said that, it is quite plausible to replace the question by an utterance where A promises B some reward upon B providing an answer to q.

Finally if agent A issues a command c to agent B in scope S, then like with questions as just mentioned, in order to understand what is going one has no need of the concept of a promise. Having said that, however, again it is plausible that A's command is replaced by a promise that A will provide some reward if B acts as if it puts command c into effect. Suggesting this replacement is plausible in particular if one has no interest in obligations that might result from the issuing of a command.

Summing up we find that promises stand out as the utterances from which all other communications can be derived. This prominence of promises results from the following four assumptions: (i) the absence of an attempt to rank different agents according to their objectivity in dealing with facts and with information provision about facts, (ii) willingness to complement the specifications of questions (commands) with specifications of rewards for adequate replies (valid realizations), (iii) non-obligationism, (iv) that obligations (especially when resulting from a command being issued) are not essential or simply not useful as a regulatory principle.

2.9.6 Promise strength

The above example indicates that once promises are made and have been noticed by an audience consisting of agents in its scope, these agents engage in a process of trust management. The strength of a promise, from the perspective of an agent in its scope, quantifies the expectation that it will be kept. The same promise may have different strength for different members of its audience, and the evolution of that strength may differ from between agents because of different experiences.

It must be stressed, however, that the promiser, the promisee, and all agents in scope are individually responsible for their management of the strength they attribute to a promise. Aggregate trust management may be used, but is is necessarily a voluptuary matter for all agents involved.

2.10 Obligations

Having explained intentions and promises, let us now try to describe the notion of obligations in the same manner. The intuitive notion of an obligation seems straightforward, but it proves to have difficult properties. We might try to think of obligations in a straightforward way, for instance: *an obligation is an intention that is perceived to be necessary by an agent*. This certainly captures some of the characteristics that we understand by the term, but it also leaves many questions unanswered: is the feeling of the necessity voluntary or forced, a matter of survival or simply an authoritarian convention?

2.10.1 Scope of obligations

An obligation, or more precisely the impact of an obligation in general, as felt by an agent within its scope, falls into the category of possible intentions. We will speak of an induced intention, the intention being achieving compliance with the obligation. The induced intention must have source, a target and a body, and the body must have a quality and a quantity. The source and target are now somewhat difficult to understand however. Unclear is further to what extent an induced intention is current, and if so if it is a commitment. Can it be indifferent, or even a deception or simply invalid?

Beyond this, we shall not attempt to define obligations more carefully. We shall merely state some assumptions about them.

2.10.2 Obligations and force

An obligation can be imposed by external conditions, e.g. by the expected behaviour of external agents, by laws threatening sanctions etc, or it can be self-imposed by codes of personal behaviour which an agent holds to be *necessary*. But this imposition suggests the action of a force which attempts to induce a commitment in another agent (or itself). An obligation is a possible intention which may or may not be current and may or may not have the status of a commitment. In any case the agent is aware of any compelling reasons to include the intention in the portfolio of commitments, either from within itself or without due to external forces.

It seems natural then to refer to the source and target of the induced intention as being the agent in which one attempts to induce the intention, and the recipient of the intention respectively. However, the source of the obligation itself might not be an agent at all, but merely a set of external conditions, norms,

experiences or other information acquired by the agent that lead to a perceived priority.

Note again that even 'forced' behaviour can be classified under the realms of (possible) intentions since all behaviour can be intended. Again, we emphasize that this does not imply that a coerced agent holds the intention that is being forced upon it. Nor does it say anything about whether the agent is able to resist the force or not, or whether it matters if an obligation is self-imposed or externally imposed.

Viewed from the perspective of an agent, the notion of an obligation immediately seems significantly more complicated than an intention or a promise and does not seem to be close to the notion of either promises or intentions.

2.10.3 Compliance and obligations

We hold that obligations are far from being a reliable tool for ensuring compliance. If a law-giver wanted to ensure the compliance of an agent, a better strategy would be to obtain a promise from the agent, and to convince it to view the intention as a *commitment* since the law-giver could never know whether the agent had indeed committed to the body of the obligation.

To study the idea of compliance further let us return now once again to promises. It is clear that promises and obligations cannot be simply related (as some promises might be deceptions) so let us explore deceptions in more detail.

2.10.4 The strength of obligations

Obligations can be assigned a measure or degree of strength from the perspective of an audience just as promises. When assessing the strength of an obligation an agent will first of all estimate the degree to which an obligation applies to its own context, rather than the degree to which the obliging entity is entitled to do so.

The divergence between promise strength and obligation strength appress strikingly when one appreciates that, unlike promise strength, obligation strength has little to do with expectations and is nearly independent of future action of any agent involved.

2.11 For and against the primacy of obligations

Promises are often treated in the context of agreements, see [She11a]. Agreements may be constructed out of mutual promises. Invariably, however promises are linked with obligations.

By the sheer weight of tradition, obligations dominate discussions of behaviour. For that reason we make an attempt to compare promises and obligations as conceptual tools for distributed systems design.

2.11.1 In favour of obligations

1. Some people might think that a promise is an obligation because it seems to create one, and might therefore be considered equivalent to that obligation. (This is a version of obligationism to which we oppose.)

2. Obligations are a well known concept from deontic logic. There is an advantage to to reducing the less well-known concept of promises to one that has been studied for more than fifty years. (This is true, but it implies no more than that promises are worth studying.)

3. Obligations have a formal status in state laws and regulations. There is no such public body of promises. (Promises are a dynamic phenomenon concurrent with autonomous action, listing promises globally and statically is not plausible.)

4. Many obligations give rise to promises which occur in the process of fulfilling an obligation. E.g. the cat must get fed while owner is on holiday, the owner is obliged to get the cat fed (by law forbidding cruelty to animals). A friend promises to help in the fulfillment of the obligation. (This is true but it does not imply that obligations are prior to promises in general.)

2.11.2 Against obligations

1. Suppose one has the concepts of promise available, and now reflects upon obligations. An agent $P_{issue:ob}$ issuing an obligation with body b_{ob} to all agents Q in a scope S, might be understood as simultaneously promising all members Q of S that $P_{issue:ob}$ will act in such a way as if P had received a promise with body B_{ob} from Q.

   This is a reasonable explanation of what may happen when issuing an obligation and it explains why obligations seem to be complex entities or events from the perspective of promises.

2. If a future promise (e.g. the promise to feed the cat in the future) is in fact a deception then this falsifies the necessity of a relationship between

promises and obligations. In other words, all intentions occurring as apparent intentions in promises cannot be induced obligations because some promises can be deceptions and these cannot be (easily) understood as induced obligations.

3. Similarly, not all promises are about future actions, so there cannot be an implied obligation capturing its essence for all promises. e.g. I promise that the cat got fed. Indeed the owner might actually be displeased that the cat was fed if it was supposed to be dieting.

4. In law, it is true that there is a dissimilarity between promises and obligations. They are quite different entities. Obligations may cause promises and promises may cause obligations, but promises have a physical reality as events in space and time, whereas obligations do not. Obligations are at a different level of abstraction altogether.

5. Promises are made on a voluntary basis. For obligation however, the concept of voluntarity is almost irrational. In any case it might be voluntary to imply an obligation on someone else, but engaging in a promise you may face an involuntary obligation or a voluntary one. Voluntarity is therefore natural for promises but is quite problematic for obligations.

6. Promise announcement constitutes positive extensions of user behaviour, whereas obligations constitute a negative constraint on the degrees of freedom of the obliged party.

If one would choose between promises and obligations, it seems abundantly clear that promises are the simpler concept. Moreover, the concept of a promise seems more natural in the technological world: since computers cannot feel ethical responsibility, the reduction to promises to obligations seems to be neither philosophically satisfactory nor technically correct. Moreover, there are some behaviours one cannot oblige (empty the ocean with a sieve). These can be promised, even if the promises are clearly invalid.

So far we have argued that promises are different, simpler and can be analyzed independently of obligations. There is one more point that is of principal practical importance. Promises are *local* constructions, whereas obligations are *non-local*. The source of a promise is localized in a single entity that has all of the information and self-control to be available to resolve conflicts and problems with multiple promises. The sources of obligations however are distributed amongst many individuals and the obliged party does not have the access to resolve the conflicts without maintaining a voluntary dialogue with all of these multiple parties.

From a practical viewpoint, obligations are simply less effective at reducing uncertainty because they tend to increase uncertainty not reduce it. Indeed, obligations can be inconsistent, but promises cannot. More precisely: consistency of promises is a matter that can be verfied at the level of sources only. Promises made by different agents cannot be inconsistent.

Preferably then one would not use obligation as a coordination principle if a mechanism based on promises can be used instead. Promises are simply more trustworthy. A collaboration based on promises works better if one has trust. In a world of obligations however, trust is meaningless because one has only a presumed outcome.

2.12 Deceptions

2.12.1 Non-intended promises

Although we have distinguished five cases for promises, commitments stand out as most important immediately followed by deceptions, with the other classes entering the picture in order to have a complete story. We add some further comments on deceptions.

Understanding deceptions (or lies) is also an important step in clarifying the relationship between intentions and promises, because it is possible for an agent to have two different intentions in play at the same time: a commitment and an announced intention (i.e. a promise) which are not compatible. Incompatibility means that striving for both intentions simultaneously is fruitless because their realizations cannot be combined.

In a deception, there is always a source and always a target and the target cannot be the same as the source, as an agent cannot (intentionally) deceive itself. Furthermore, we maintain that the target of a deception must be in scope, so there must be a physical documentation and hence a deception necessarily involves a promise and not merely an intention.

Definition 12 (Deception) *A deception consists of two intentions: a documented intention (i.e. a promise) and a non-documented commitment, which are incompatible.*

The non-documented commitment will also be called the hidden intention.

In a deception the hidden intention is more important than the witnessed one one and we might refer to it as the dominant intention. This simply expresses that it is a commitment while the promise contains merely a “possible intention”. It is the *real* intention of the agent (“intended intention”), while the intention in

the promise can merely be described as *non-real*. If the dominant intention should be rescinded, a deception will revert to being a promise, but this is only known to the source.

2.12.2 Non-deceptions

A deception is the augmentation of a promise with an *incompatible* intention. We should like a name for the augmentation of a promise with a positive intention. We might call this a promised commitment, or intended promise. From these slightly strained terms, we can now appreciate why the concept of a promise is in fact so important. A promise is simply a promise (the documentation of an intention), regardless of what lies behind it. Any internal priorities or considerations are hidden from the view of other agents and cannot be observed. Thus, promises are an independently important concept because we can (indeed must) talk about promises without discussing the basis on which they are made.

When a promise is made, we are neither required nor able to confront the truth or falsity of the promise. Indeed, as soon as we ask such questions, new issues such as trust and a plethora of other subjective issues come into play. Such issues are probably un-resolvable in a logical sense. However, what we assume is here that no matter how trustworthy a promise might be, it can increase or decrease our certainty of a promised outcome and thus it bears an *influence*. The matter of assessing the promise can be very complicated and uncertain and we shall not attempt to discuss this here in any depth.

2.13 Ranking of intentionality

The foregoing discussion of deceptions suggests the existence of a ranking function which induces a partial ordering onto the intentions that are referred to or in play at any given moment. There are intentions one intends to invoke (i.e. that one commits to), intentions one prefers, intentions one feels obliged to intend, and finally deceptions which one intends to not honour the intention documented for a wider audience.

2.14 Assessing promises kept

The notion of whether promises are kept or not is central to their sustained usefulness in society, thus we need to make mention of how this comes about in a theory of promises. It would be easy to go overboard and delve into the complexities of observation and measurement to provide a satisfactory answer but

Promise	Representation	Scope	Assessment
Fed the cat	Speech act	Those who heard the promise	Either did or did not.
Credit card accepted	Action	Visitors to the store	Either did or did not.
Response in 24 hours	Contract	Signing parties	Reply received in time or not.
$var = value$	Source code	Readers of the code / compiler	Syntax ok. Value in range. etc
$var = value$	Object code	Execution engine	Ok to execute, exception, etc.

Table 2.1. Promise assessments

that is not in the spirit of this paper. We seek instead a simpler notion which is at the same level of abstraction as the concepts of promise and intention that we have introduced thus far. We call this the concept of *assessment*.

Definition 13 (Assessment) *An assessment is a subjective statement made by an agent about whether the intentions of itself or of another agent were fulfilled.*

Our notion of an assessment is more generic and less quantitative than a verification. It is both subjective and not *a priori* linked to observation. In a static theory of promises and intentions the existence of intentions as well as the value of assessments is linked to state parameters like time. Thus, for an intention of agent A, in existence at time t, it may be the case that agent B's assessment, made at time $t' > t$ is positive (or negative).

At this level of description, we need not say any more about it than this. The reality of whether promises have been kept through specific actions is neither here nor there in the world of politics and to some extent economics. What is important is how a witness to the promise assesses the outcome of the promise. Such an assessor may or may not feel obliged to assess an outcome in a particular way, might promise to conform to certain criteria, and so on. What matters is only the assessment, which might or might not be rationally obtained. We believe that this is a fair model of the world in which we live.

An assessment is a supposed outcome relative to some method of assessment. Assessment involves a variety of possible routes to inference, i.e. there are different kinds of assessment. This is a subjective issue, but this should not be viewed as a weakness of our theory: it is a true feature of the subjective nature of individual assessment.

Some example assessments are shown in table 2.1. We see that assessments are quite sensitive to physical representation of the promise. Once again the notion of representation (or documentation) is a key to the importance of a promise as a concept.

2.15 The subjective value of promises

Promises are valuable to agents, because they help reduce uncertainty and because their outcomes could be beneficial if they become certain. Because certainty is key, a promise is worth nothing unless there is trust. Zero trust makes promises worthless. Trust might be based on a history of keeping promises or, in our terminology, on a history of positive assessments about a succession of promises. So there is a symmetry between trust and promises that must be broken to solve the dilemma.

If there is trust, a promise about future behaviour does not need to be perceived as an obligation on the promiser but as an indication that best effort will be respected. If a given agent X does not trust the promiser however, it might assume that the promise implies an obligation on the promiser. This perception of obligation by X does not of course imply an obligation perceived by the promiser. There is a fundamental subjectivity in these perceptions.

The value of a promise is an expectation of the eventual benefit. Suppose, then that A promises B 400 dollars per year. B promises to wash A's windows at this price. Both are satisfied with the value they get from this arrangement and prefer not to question it too much as this could unleash all kinds of consequences. Observer C can see that the values are quite mismatched, or that A is getting a poor deal by its judgement, but C also cannot deny that the relationship is stable because both A and B are happy.

The value of promises may be questioned by those who consider promises as a concept secondary to obligations. If one views obligations as the primary concept, the value or importance of promises unavoidably shrinks. We shall now survey advantages and disadvantages of obligations as an alternative cornerstone of a theory of promises. The discussion will be somewhat asymmetric because we will not base our comparison on a proper definition of the concept of an obligation (which we cannot fathom). Suffice it to say that for some agent A an obligation is an intention ("possible intention"), which has been elevated to the status of an obligation, whatever the consequences of this status may be.

2.16 Semantics and dynamics of agents and promises

Throughout this work, we shall be confronted with issues from two different perspectives: dynamical and semantic[5]. The dynamics of agents and their promises refer to actual patterns of behaviour, describable in terms of space and time and the agents' characteristic variables. The semantics, on the other hand, refer to agents' interpretations of one another's intentional behaviour. A theory of promises is not possible without both of these aspects.

2.17 Summary

Without attempting to suggest applications in any field, we have argued for the usefulness of promises as an independent and practical concept, whether in philosophy, economics or technology. We have compared promises to the more usual idea of obligations and have concluded unequivocally that promises are a simpler theoretical notion and a more practical tool than obligations in the reduction of an agent's uncertainty about the behaviour of other agents.

We show that intentions, promises and commitments can be explained in the absence of an understanding of obligations. Furthermore, although it seems to be a common view that obligations are a more primitive concept than promises, our paper suggests the contrary. Promises need not be viewed merely as proxies for obligations; if promises give rise to obligations, this can in fact be studied purely in an exposition based on promises, intentions and commitments. Indeed more often than not promises are made by agents who would not be able to explain the extent to which their promises might lead to obligations or not, or to what extent such obligations would be more significant than the promises from which they arose.

Notes

[1]It is possible to promise that something happened in the past, which is difficult to understand from a moral perspective. Rather such promises are about strengthening expectations and trust.

[2]Why are obligations discussed so much and promises so little? The reasons for this are interesting in their own right. We can only speculate on them here. Probably the concept of obligation is tied to the way in which humans govern, top down. The law, starting with perhaps the Magna Carta, set a norm for centralized governance by a few powerful individuals who exerted organization or influence over a large number of others. As time has passed, desirable behaviour has become the norm and attention naturally refocuses on individuals' voluntary cooperation rather than top down force and obligation.

[3]The phrase 'exact science' is of course pure nonsense. As Hume pointed out, there are two kinds of knowledge: provable assertions about simplified systems where all assumptions are laid out (i.e. a fictitious world), and unavoidably approximate or even ad hoc observations and derivations of the real world. In both cases there is is uncertainty.

[4]It has been common to start with the *rationality* of agents (decision theory), or the capabilities of agents as a measure of their independence from one another (multi-agent systems). Game theory also deals with the issue of incomplete information, but none of these models lives up to the challenge – they inevitably find a way around what they perceive as limitations, or never consider the full implications of the autonomy. That is the task that promise theory sets itself.

[5]For a discussion of these topics, see In Search of Certainty by Mark Burgess [Bur13].

3
Formalizing Promises

To make a theory of promises and the world in which they live, we need a formulation that abstracts away non-essential details, but which retains significant expressive power. We therefore turn to a specific model of promises that we call *micropromises* or μ-promises—in order to emphasize that this is a particular formalization, and one might still conceivably make other models; however, we shall drop this title and simple speak of promises for the most part. For the remainder of the book, it will be understood that we are talking about a particular model, and we shall describe some basic nomenclature and conventions.

In this chapter, we attempt to set down basic definitions and rules by which to manipulate them. These rules maintain the assumptions of the model.

3.1 Agents and ensembles

Micropromises (henceforth simply promises) are given and received by entities called *agents*, which may be human, computer or completely inanimate. They are the agencies of decision-making and of change that comprise a system. They are atomic parts.

Agents play many roles, as the givers and receivers of promises. In the role of *promisers*, agents may make promises about information, behaviours, and so on. In the role of *promisee* they are the recipients of such promises.

We shall suppose that the internal workings of agents are always hidden, unless we choose to reveal their internal structure by breaking a single agent up into multiple agents; hence, agent structure is not relevant to a promise, nor can it be be known except by indirect inference. An agent is mainly a convenient node or origin for promises. All that is known about promises is advertised by the promiser; this might allow other agents to speculate about the agent's internal

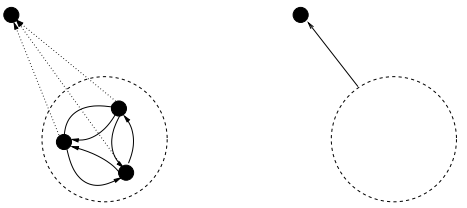


Fig. 3.1. Agents might have internal structure (left), revealed by differentiated promises, but that structure might be presented as a single agent package (right). The atomicity of the agents is therefore not as fundamental as the atomicity of the promises they make.

structure (see fig. 3.1), but only from the viewpoint of incomplete information.

If a single promise can be logically decomposed into a number of more elementary promises, then it is possible that those elementary promises in fact exist and could be given by different internal parts of the agent that have not been revealed. An agent is thus a black box interface for behavioural declarations.

We shall refer to a collection of agents as an *ensemble*, anticipating the language of statistical physics. The list in table 3.1 are all agents in the promise theoretic sense.

Agent	Promise body
Electron	Negative electric charge
Web server	Deliver html pages
Can of soup	Contains soup
Bus	Transport to location on front sign
Human	Obey the law of the land.

Table 3.1. Some example promiser agents.

Formally, we assume that agents implicitly assume their own survival for the duration of the promises they make.

3.2 μ-Promises defined

Consider a general set of N agents A_i, where $i = 1, \ldots, N$. We denote agents by upper-case Roman letters, and shall often use nicknames A_s for promise-sender or giver, a_r for promise receiver and A_t for third parties, to assist the discussion.

Definition 14 (Promise) *A (micro)promise is an autonomous declaration of possible behaviour from a* promiser *or promise giver, to one or more* promisees *or recipients. These recipients have yet to verify the behaviour.*

Each promise contains a promise body b that explains the content of the promise.

The scope *of the promise is the set of agents who have knowledge of the promise that has been made; this may or may not include the promisees.*

We denote a promise from agent A_s to an agent A_r, with body b by:

$$A_s \xrightarrow{b} A_r \tag{3.1}$$

We consider promises to be *autonomous* declarations; in other words, the making of a promise is not someting that can be imposed on an agent by external forces[6]. If agents are humans, we may consider them to be made voluntarily; if they are machines, they are made as part of their design. This will turn out to be the most elementary point of view, and therefore the most fundamental and useful model of promises, although we shall qualify this below in section 3.4.

We say that a promise refers to unverified behaviour, because one does not promise something that is already known. Such a promise would be redundant. Thus one might promise to pick up someone from the airport, about an event in the future, or promise to have finished an assignment in the past, but the key point is that the recipient of the promise has yet to verify the outcome.

A promise is given typically to one or more individuals. This might simply be oneself or it could be the entire world, but often it is limited to a select few. The promise might nevertheless become known by a larger audience, which we refer to as the *scope* of the promise[7].

Finally, the content of the promise is referred to as the body of the promise, by analogy with the body of a document, or the body of a contract. The body explains the extent to which the making of the promise either *will constrain* or *has constrained* the behaviour of the agent making it.

3.2.1 Body types and constraints

The body of a promise describes its details, in the sense of the body of a document or of a contract.

Each promise body b contains:

1. A name or label $\Lambda(b)$ that uniquely identifies the promise from similar of dissimilar promises to the agent.

2. A type $\tau(b)$ that explains the nature of the promise to the agent.

3. An explicit constraint $\chi(b)$ on the affected state of the agent.

It is important for promises to have names so that they are distinguishable. This enables both provenance tracking and distinction of causation. Although the promised outcome can never depend on the names of promises, it is often helpful to have a unique name when promise bodies might overlap. Promises that are indistinguishable have no purpose because, as we shall show, they are mathematically idempotent. Thus identical promises with the same name (or without a name) are simply a reiteration of the same promise. By using names, one could artificially distinguish between identical promises for the sake of a larger coordination of process.

Agents too might be indistinguishable, but agents are not idempotent, as they represent the real entities of the system.

The promise body constraint $\chi(b)$ might not obviously be a constraint. For instance, if you promise to brush your teeth, you are really promising one course of action out of all the many possible outcomes of your behaviour of type 'state of teeth'. Of course, there might be other ways to classify types. A natural way is to classify promise types by the kinds of entities that are affected. This is up to each agent, as an autonomous entity to decide. We might then define the promise type τ = 'teeth' with set of possible outcomes $o \in \{white, clean\}$. Then the promise to brush your teeth is involves the the desired outcome constraint $\chi(o) : o = white$.

Promise types distinguish the qualitative differences between promises, and additional constraint attributes distinguish the quantitative differences. For each promise body b, there is another promise body $\neg b$ which represents the negation of b. We shall assume that $\tau(\neg b) = \tau(b)$ and that $\neg\neg b = b$. The negation of b refers to the deliberate act of not performing b, or what ever is the complementary action of type $\tau(b)$. Thus we shall often write the promise body as a pair $b \sim (\tau(b), \chi(b))$.

3.3 Knowledge and information relativity

The fact that each agent is an isolated autonomous entity means that it observes and stores information in a unique perspective. Each agent's view of the world of other agents is *a priori* unique.

If two agents attempt to compare their information or world views on a particular subject, they might or might not agree. This is called agent relativity[8]. Relativity of agents in a discrete network will lead to consider the many-worlds

scenarios of Leibniz, Kripke and Everett. Every agent experiences, for all intents and purposes, a completely independent branch of reality, until is begins to cooperate and exchange information with other agents.

What do we mean by knowledge? Knowledge is acquired by an agent when using information repeatedly. Information becomes knowledge when it is contextualized through repeated interaction into recognizable behavioural patterns[9]. Knowledge is recognition. It seems reasonable to assume that agents' own internal data values represent knowledge; however, acquiring knowledge from other external sources requires some commitment to interact.

Thus, we typically think of knowledge as referring to recognizable, repeated phenomena. Promises that are refer to one-off actions can still be regarded as knowledge, even though there is no repeated action involved in keeping them, because the information about the intent itself can be revisited or is available for repeated inspection. Thus we shall refer to the information about promises as knowledge, even though it strictly only becomes knowledge through repeated usage.

Agents can learn and store knowledge. Indeed, there is information that could potentially be called knowledge in the bodies of the promises they make. As other agents repeatedly assess the promises of a given agent, the promises acquire the status of knowledge about the agent.

Promises are a documentation of intent. It therefore inevitable that such knowledge might be passed on to other agents, spreading the knowledge about a promise to others. Indeed, this is exactly what we mean by scope (see section 2.8).

3.3.1 Denoting information about promises

Agents need to refer to the promises they make. Since knowledge of promise, by any agent, could itself be the subject of a promise (e.g. I promise that X told me about her promise to be faithful to Y), there must be promise bodies that contain the knowledge of other promises, so we shall need a standard notation to represent this. It might be sufficient to write the promises themselves $\pi_1, \pi_2, \ldots$ etc, or even their bodies $b_1, b_2, \ldots$. for such knowledge. However, we shall try to emphasize the idea that knowledge is only a function of information by using additional notation to say that we are not treating the promise as active but merely as the subject of commentary.

The notation π_i ($i = 0, 1, 2, 3, \ldots$) labels complete promises including the promiser, a list of promisees, and the body with all of its details. We shall write the inactive definition of a promise as by $\psi(\pi)$ to distinguish this. This is meant

to denote a distinction between a depiction of the promise $\psi(\pi)$ and the real thing π, just as there is a different between a newspaper article about a promise and the promise itself.

Definition 15 (Knowable information about promises) *The promise body for talking about a promise π is denoted by by $\psi(\pi)$.*

If we wish use this as part of a promise, e.g. I promise (π_1) that I promised something else (π_2), then the first promise π_1 has body 'promise description' $b_1 = \psi(\pi_2)$. In other words, promise π_1 constrains the body of the promise to simply be a verbatim description of the promise π_2, like packaging in an envelope.

We use the notation $\psi(\pi)$ rather than simply π to distinguish between a promise that is to be acted upon, and a description of the promise that is only meant to be 'known'. Formally these objects have different 'types', i.e. a promise is not the same as a description of it, even though the promise description must exist for an agent to know about it.

3.3.2 Denoting promise scope

Knowledge about promises is clearly at the heart of the notion of promise *scope* (see section 2.8). The abilty for a promise to drive expectations relies on it being known. The scope of a promise is the collection of agents that have received information about its existence.

Definition 16 (Scope) *We denote the scope of a promise by a set of agents σ, with whom information $\psi(\pi)$ is shared. This scope may optionally be written under the promise arrow.*

$$A_1 \xrightarrow[\{\sigma\}]{b} A_2 \tag{3.2}$$

The default scope is the promiser plus the promisee:

$$A_1 \xrightarrow{b} A_2 \equiv A_1 \xrightarrow[\{A_1,A_2\}]{b} A_2 \tag{3.3}$$

3.4 Classification of kinds of promises

Below we describe four kinds of promises, in stages of complexity. We shall conclude that only a single type of promise is needed to explain all the behaviour,

by assuming the autonomy of the agents. We show that promises of the first kind are an elementary and even fundamental kind of object.

The generic form of these is:

$$A[B] \xrightarrow{b} C[D] \tag{3.4}$$

where A is the promiser, C is a promisee or intended recipient of the communication, B is the agent responsible for keeping he promise, and D is the party affected by the outcome.

3.4.1 Promises of the first kind

The simplest form of a promise is the one we have used up to this point. We write it:

$$\pi : S \xrightarrow{b} R \tag{3.5}$$

where π is its name, and the triplet $\langle S, b, R \rangle$ represent the sender agent S, the promise body b, and the recipient agent R.

It is assumed that this is a local and autonomously made promise. It is equivalent to a promise of the fourth kind, described by the notation:

$$\pi : S[S] \xrightarrow{b} R[R]. \tag{3.6}$$

i.e. S promises b to R.

3.4.2 Promises of the second kind

A promise of the second kind allows obligation:

$$\pi : S[T] \xrightarrow{b} R \tag{3.7}$$

i.e. S promises R that it will oblige T to act as if it had promised b to R. If T is autonomous, this is forbidden and has no influence on T.

Note that, in this description, we do not distinguish between obliging and forcing. This is a subtley yet to be resolved. Ultimately, the notion of autonomous intention would have to be sacrified in the presence of overwhelming force.

3.4.3 Promises of the third kind

A promise of the third kind is autonomously given but allows indirection.

$$\pi : S \xrightarrow{b} R[T] \tag{3.8}$$

i.e. S promises to R that S will do b for T. Promises of the third kind hold a special significance for transmission of knowledge and promise *scope* (see section 3.3).

Consider a promise by S to R about a promise with body b to a third agent T. We could write this in two ways. We use the notation $\Psi(b)$ for a promise body that describes the knowledge about promises bodies of type b. So we could write this as a promise of the third kind directly:

$$\pi_3 : S \xrightarrow{b} R[T] \tag{3.9}$$

which would furninsh R with complete information about b and its recipient T. But we can also write this as a promise of the first kind about the knowledge $\psi(\cdot)$ of the original promise:

$$\pi_1 : S \xrightarrow{b} [T] \tag{3.10}$$

$$\pi_2 : S+ \xrightarrow{\pi_1} R \tag{3.11}$$

Now, since π_1 contains the full description of the promise (which may also be written as a triplet $\pi_1 = \langle A, b, T \rangle$), the same information has been conveyed to another agent. This is a promise about a promise. We clearly cannot exclude the names of agents from the promise body in general, and this indicates that .

3.4.4 Promises of the fourth and most general kind

A promise of the fourth kind is the most generic form:

$$\pi : S[T] \xrightarrow{b} D[U] \tag{3.12}$$

i.e. S promises D that T will act as if it had promised b to U. Note that, in this case S, T cannot be considered an autonomous agent, for if T had been autonomous, this would be forbidden as S would have no influence of T. However, once again, we can express the intention here using purely voluntary cooperation, using promises of the first kind – this time, it involves a promise about a

promise:

$$\pi_1 : S \xrightarrow{\pi_3} D \tag{3.13}$$

$$\pi_2 : S \xrightarrow{\pi_4} D \tag{3.14}$$

$$\pi_3 : T \xrightarrow{U(b)} S, \tag{3.15}$$

$$\pi_4 : T \xrightarrow{b} U, \tag{3.16}$$

i.e. S promises D that it knows of a promise π_3 (which has body $U(b)$) made from T to S to acquire information about body b. S knows this information according to eqn. 3.12. Moreover, S knows of another promise π_4 made by T to U that it intends to use this knowledge of b to make a promise itself with this body to U. This completes the construction of a replacement for the promise statement 3.12 by promises of the first kind.

3.4.5 Elementarity of promises of the first kind

From the preceding list, it appears as though we can construct facsimiles of all of the promise kinds, using only promises of the first kind. This means that we can always re-express obligations in terms of promises of the first kind. This motivates the study of these elementary promises in some depth.

Whether or not this is desirable can be discussed, but regardless of what one thinks of the description, the utility of such a device would be considerable. Assuming that no agent can wield irresistable force in all matters, even the expression of an obligation cannot lead to any certainty of outcome in the matter of the body b. It offers a uniquely pragmatic point of view then to frame the problem in a language that is open to consider its own limitations[10].

Theorem 1 (Projection theorem for promises of the n-th kind) *Let $\pi^{(n)}$ be a promise of the n-th kind, and A_1, A_2, A_3, A_4 be agents that are not necessarily, distinct. It is always possible to substitute for $\pi^{(n)}$ a finite number of promises of the first kind such that the resulting assurances are the same.*

The proof for $n = 4$ is construction explicitly in (3.16).

Henceforth, the promises in this book shall only refer to those of the first kind.

3.5 Promise graphs

The formulation of μ-promises has the obvious characteristics of a network, or in graph theoretical terms a so-called *directed graph* (a network of arrows). This is not a novel or unusual construction[11]; many phenomena form such networks.

However, its commonality is a powerful identification of its ubiquity, and this feature of promises will allow us to draw in many important insights that have been made about networks in later chapters. For example, directedness in graphs display the intricacies of *causation*: the ordering of multi-agent phenomena.

When a single agents makes a collection of promises to other agents, some of these can be simplified or replaced by a single promise. There can also be cases in which we attribute special meaning (semantics) to particular combinations of promises, thus we begin by discussing the basics of composition.

3.6 Knowledge transmission and dissemination

Passing observables from agent to agent leads to a spread of information, and it can also lead to the distortion of information. How far does this loss of fidelity matter?

We have not discussed the scope of a promise since introducing the notion informally in the introduction. However, if we take the notion of agent autonomy seriously, we must address the question of how knowledge (say of a promise) becomes known to an audience of other agents.

3.6.1 Defintion of knowledge

We shall equate knowledge with information that has been assimilated by agents (see definition below) to reflect the timelessness of promises. We wish to emphasize that a promise is a continuously maintained state, not a mere data transaction, and thus keeping a promise implies the kind of relationship that is suggested by speaking about 'knowledge' rather than information.

Definition 17 (Agent Knowledge) *Let $K^{\gamma}_{A_i}$ be a set of data values that are internal to agent node A_i. We call $K^{\gamma}_{A_i}$ the knowledge of agent A_i about an observable value γ. This knowledge is not available to any other agent, unless its value is promised to them.*

We use a notation $K^{\gamma}, K^{\gamma}_1, K^{\gamma}_2$ as follows. Let $\mathcal{K}$ be the set of all possible variables or subjects that can be known, and consider knowledge about a specific set of values or subject $K^{\gamma} \in \mathcal{K}$. We use K^{γ}_1 to mean agent A_1's knowledge of this subject γ, and K^{γ}_2 to be A_2's knowledge about K^{γ} etc. Thus different agents (i) can have different knowledge about the same subject (γ). We shall use this, in particular, to discuss agents' varying type schemas.

3.6.2 Distributed consistency of knowledge

Definition 18 (Consistent Knowledge) *Two agents A_s and A_r, with knowledge K_s^γ and K_r^γ, can claim to have consistent knowledge about k^γ iff they each can verify the equality of a subset of their knowledge $k_s^\gamma = k_r^\gamma$ where $k_s^\gamma \subseteq K_s^\gamma$ and $k_r^\gamma \subseteq K_r^\gamma$.*

To simplify notation henceforth, we pick a single subject and suppress the γ superscripts.

Theorem 2 (Consistent Knowledge) *Let A_s and A_r be any two agent nodes, and let $\pi : k$ be a promise body that implies communicating knowledge k. Nodes A_s and A_r have consistent knowledge k iff*

$$A_s \xrightarrow{+k_s} A_r \tag{3.17}$$

$$A_r \xrightarrow{-k_s} A_s \tag{3.18}$$

$$A_r \xrightarrow{+k_r} A_s \tag{3.19}$$

$$A_s \xrightarrow{-k_r} r \tag{3.20}$$

$$A_s \xrightarrow{(k_s = f(k_r, k_s))} A_r \tag{3.21}$$

$$A_r \xrightarrow{(k_r = f(k_r, k_s))} A_s \tag{3.22}$$

where $f(a, b)$ is some function of the original values that must be agreed upon.

Proof: By definition 17 the knowledge k_s of any agent S is only available to another agent node A_r iff its value is promised as in eqns. (3.17) and (3.19). The knowledge is not certainly received until the agents promise to accept the promised values, as in eqns. (3.18) and (3.20). Now both agents know each others' values. To make their knowledge consistent, by definition 18 they must now promise each other to make their values equal. This can be done through an intermediary function $f(a, b)$ which signifies that an equilibrium is formed by some unspecified concensus. ∎

Consistency of knowledge is a strong concept which requires the notion of verification; at the same time it is clear that knowledge might be both available and consistent in a service oriented network, even if one cannot actually verify that consistency, e.g. if it is disseminated from a single source.

3.6.3 Assimilation of knowledge

Let ψ_s be the knowledge of agent A_s. Knowledge ψ_s is said to be relayed from a source A_s to a receiver agent A_r iff

$$\begin{aligned} A_s &\xrightarrow{\psi_s} A_r \\ A_r &\xrightarrow{U(\psi_s)} A_s \end{aligned} \tag{3.23}$$

The fact that knowledge has been passed on and received does not mean that it has been *assimilated* by the agent, i.e. that it will replace its own knowledge about ψ with this new version. For example, in peer to peer systems, such as media distribution, relay stations, routers, data are frequently relayed in this way without necessarily being retained. A new version of a song or movie could be passed on to others without replacing a local authoritative version used by the local system.

Definition 19 (Assimilated knowledge) *Knowledge ψ_s is said to be assimilated by a receiver agent A_r from a source A_s iff*

$$\begin{aligned} A_s &\xrightarrow{\psi_s} A_r \\ A_r &\xrightarrow{U(\psi_s)} A_1 \\ A_r &\xrightarrow{(\psi_r = \psi_s)} A_2 \end{aligned} \tag{3.24}$$

where A_1, A_2 could be some unspecified agents. We shall assume here that $A_1 = A_2 = A_s$ for simplicity.

An agent does not know ψ unless it is either the source of the knowledge, or it has promised to accept and assimilate the knowledge from the source. It does not matter to whom the promises to use and assimilate the data are made, as long as they are kept, though it seems natural to make the promises to A_s so that A_s can verify the transmission.

3.6.4 Integrity of information through intermediaries

Knowledge ψ can be passed on from agent to agent with integrity, but we must distinguish between agents that end up with a different picture of the information as a result of their local policies for receiving and relaying their knowledge.

1. Accepted from a source, ignored and passed on to a third party intact.

$$\begin{aligned} A_1 &\xrightarrow{\psi_1} A_2 \\ A_2 &\xrightarrow{U(\psi_1)} A_1 \\ A_2 &\xrightarrow{\psi_1} A_3 \end{aligned} \tag{3.25}$$

   Note that the agent A_2 does not assimilate the knowledge here by making its own version ψ_2 equals to ψ_1, it merely passes on the value as hearsay.

2. Accepted from a source, ignored and local knowledge is then passed on to a third party instead.

$$\begin{aligned} A_1 &\xrightarrow{\psi_1} A_2 \\ A_2 &\xrightarrow{U(\psi_1)} A_1 \\ A_2 &\xrightarrow{\psi_2} A_3 \end{aligned} \tag{3.26}$$

   Here the agent A_2 accepts the information but instead of passing it on, passes on its own version. The source does not know that A_2 has not relayed its data with integrity.

3. Accepted and assimilated by an agent before being passed on to a third party with assurances of integrity.

$$\begin{aligned} A_1 &\xrightarrow{\psi_1} A_2 \\ A_2 &\xrightarrow{U(\psi_1)} A_1 \\ A_2 &\xrightarrow{\psi_2=\psi_1} A_1 \\ A_2 &\xrightarrow{\psi_2=\psi_1} A_3 \\ A_2 &\xrightarrow{\psi_2/\psi_1} A_3 \end{aligned} \tag{3.27}$$

   A_2 uses the data from A_1, assimilates it ($\psi_2 = \psi_1$) and promises to pass it on (conditionally ψ_2/ψ_1) if it receives ψ_1. It also promises to both involved parties to assimilate the knowledge. Only in this case does the knowledge ψ become common knowledge if one continues this chain.

The first two situations are indistinguishable by the receiving agents. In the final case the promises to make $\psi_1 = \psi_2$ provide the information that guarantees consistency of knowledge throughout the scope of this pattern. We can now define scope.

3.6.5 Laws about information integrity

From the foregoing discussion, it seems reasonable to define as something that is passed on directly from one agent to another. As soon as s third part is involved, we cannot say anything about the integrity of the information.

Theorem 3 (Information integrity) *The integrity of data that are relayed by an intermediate agent cannot be assumed without direct comparison from the original source...*

Any agent that has touched the data, means we cannot NOT say that the information has changed, because the agent is autonomous and we don't necessarily know what other promises it has made.

3.6.6 Uniformity and common knowledge

Definition 20 (Scope of common knowledge) *Let S be a set of source agents with consistent knowledge ψ, and let the set $\mathcal{A}(X, \psi)$ mean the set of nodes that have assimilated knowledge from a set of agents X. The scope $\mathcal{S}(\psi)$ of ψ is the union of S with all agents that have assimilated knowledge originating from S, i.e.*

$$\mathcal{S}(\psi) = S \ \cup \ \mathcal{A}(S \cup \mathcal{S}(\psi), \psi) \tag{3.28}$$

The simplest way to achieve common knowledge is to have all agents assimilate the knowledge directly from a single source agent. This minimizes the potential uncertainties, and the source itself can be the judge of whether the appropriate promises have been given by all agents mediating in the interaction. The scope is then simply a cooperative promise group in sense of ref. [BFa]. This is the common understanding of how a network *directory service* works. Although simplest, the centralized source model is not better than one in which data are passed on epidemically from peer to peer. The problem then is simply in knowing the boundaries of scope.

From these definitions, we can see that agents can have consistent knowledge from an authoritative source, either with or without centralization, and scope can be controlled using promise theory to describe both OO and SOA.

3.7 The context in which a promise is made

Promises are not made in a vacuum. The environment surrounding promises is summarized by the list points in table 3.2. Similarly, the context surrounding

verification of promises kept might be classified in a similar way, see table 3.3. Whenever we make promises, either implicitly or explicitly, there is a challenge to document this context.

If the natural laws of the universe are framed as promises, made by the entities that hold to them, then there is a clear and fixed context for most of these. But in the world of human creativity, and technology, the semantics of promises are freely imagined for creative purposes. This means we can assume less about mutable human promises than we can about immutable natural behaviour. This is an additional challenge to be faced by a theory of promises.

Component	Description
Promise	What outcome we intend
Algorithm	How a promise is kept
Motive	Why a promise is made
Context	Where, when a promise is made (patterns)

Table 3.2. The context of a promise

Component	Description
Outcome	What was measured during the assessment?
Context	Where was the sample taken?
Algorithm	How was the sample taken?
Motive	Why
Expectation	What were we expecting to find?

Table 3.3. The context of an assessment

3.8 Promises involving groups, anonymous and wildcard agents

We can define notation for a promise from one set of nodes to another set of nodes using set curly-braces:

$$\{A_i\} \overset{b}{\rightarrow} \{A_j\} \tag{3.29}$$

i.e. the set of agents $\{A_i\}$ promises b to another set $\{A_j\}$ of agents. Some caution is required in interpreting this notation, since the agents are autonomous, because we do not know, in advance, whether the 'promiser'-nodes will keep the same promise to the 'promisees', unless they have agreed to do so (see proposition (3)).

We can now suppose what happens when refering to third-party agents in the body of a promise. In some circumstances, this can apparently lead to contradictions. As a general rule, it seems to be in the interest of clarity to avoid this where possible.

The general promise notation above is designed to avoid the problem of referring to other agents in the body of a promse, but to make some promises we need to talk about third parties. Let us denote:

- Unspecified agents "U"
- Anonymous agents "$U^?$".

Ensembles of these can be denoted with braces, e.g. $\{U\}$ would be an ensemble of unspecified agents.

We can write

$$U^?(\{A\}_1) \xrightarrow{b} U^?(\{A\}_2) \tag{3.30}$$

to mean one anonymous agent from the ensemble $\{A\}_1$ makes a promise to another anonymous agent from the ensemble $\{A\}_2$.

Examples:

1. $(U^?) \xrightarrow{b} (U)$: someone promises (anonymously) that b will happen to someone. For example, a death threat.

   $\mathrm{Terrorist}[U] \xrightarrow{\mathrm{attack}} \mathrm{President}[U]$: Terrorist promises President that one of his operatives will attack someone.

   $\mathrm{Terrorist}[?] \xrightarrow{\mathrm{attack}} \mathrm{President}[U]$: Terrorist promises President that an anonymous individual will attack someone.

2. Let T be a set of trusted parties, then

   $$\mathrm{Parent}[U(\{T\})] \xrightarrow{\mathrm{collect-from-school}} \mathrm{Child} \tag{3.31}$$

   means, a parent promises its child that one of the trusted set will collect the child from school.

3. Let T be a set of trusted parties, then

   $$\mathrm{Child}[U^?] \xrightarrow{\neg\mathrm{U(collect-from-school)}} \mathrm{Parent} \tag{3.32}$$

   The child promises its parent that it will not accept lifts from strangers.

$$\text{Child}[U(-\{T\})] \xrightarrow{\neg\text{U(collect}-\text{from}-\text{school)}} \text{Parent} \qquad (3.33)$$

The child promises its parent that it will not accept lifts from someone who is not in the list of authorized minders..

3.9 Particular kinds of promises

Certain kinds of promises stand out as special cases and we shall need to refer to these concepts in daily usage. We define some of these here for reference.

3.9.1 Exact and inexact promises

Definition 21 (Exact and inexact promises) *A promise is exact if it allows no residual degrees of freedom, else it is inexact.*

A promise $A_1 \xrightarrow{b} A_2$ is inexact if the constraint $\chi(b)$ has residual degrees of freedom. i.e. if it is not a complete and unambiguous behavioural specification.

Example 1 *A promise body constraining a variable q such that $q = 5$ is an exact promise specification, while $1 < q < 5$ is inexact. The same principle applies to the possible outcome of a promise, however the actual outcome is naturally exact in each measurement.*

Example 2 *When electronic components promise their capabilities within certain tolerances, these are inexact promises, e.g. a resistor that promises a rating of 100 Ohms, plus or minus 5 percent.*

3.9.2 Superfluous or empty promises

Several kinds of promise body are useless. A case of this is clearly provided by empty promise.

Definition 22 (Empty promise) *An empty promise is one whose promise body contains no type or constraint of any kind.*

Empty promises can be assumed to be kept at all times, since there is nothing to verify. Note that an empty promise is not the same as making a promise that something else should be empty, or that something should be zero, since both those cases make a constraint on the content of the 'something'.

Another class of promises is about things that are inevitable.

Definition 23 (Promise bodies that are necessarily kept) *Let $K = \langle \tau, \kappa \rangle$ be the collection of promise bodies b that are necessarily kept. K is indexed by type, just as promises are.*

$$b \in K \Rightarrow \langle \tau, \chi \rangle \in \langle \tau, \kappa \rangle \tag{3.34}$$

Example 3 *A promise that the promiser exists would be necessarily kept. A promise that al recipients will receive a promise would be inevitable.*

Promises that are necessarily kept are remniscent of modal logics of necessity. Since promises are in general voluntary or incidental claims, there are few promises that can be considered necessary or inevitable.

3.9.3 Polarity ± of promises

We have already described how promise bodies may be classified into distinct types that represent the different subjects and issues about which one might make promises. Even without knowing these types, we can make a broad classification of promises into two major classes that follow from the directed nature of the promise graph.

On a graph, promises are represented as incoming or outgoing arrows, pointing from promiser to promisee. It follows from the autonomy of agents in a system, that for every promise to *give* something from A_1 to A_2, there is a possible counter-promise to *accept* or *receive*, which points in the opposite direction. Such pairs of promises form the basis of *negotiations*, and a pair of promises back-to-back are said to form a *binding*.

We shall denote these promises with a $\pm$ signs, as below:

- \+ A promise of the form $A_1 \xrightarrow{+b} A_2$ is an offer to give
- \- A promise of the form $A_2 \xrightarrow{-b} A_1$ is an acceptance to receive

If an agent offers b_1 and another agent accepts b_2, the possible overlap $b_1 \cap b_2$ is called the effective action of the promise (see section 7.4).

The significance of these two classes is fundamental and reveals a basic duality of viewpoints when formulating collective behaviour in terms of promises (see section 5.6), not unlike the way that positive and negative charges lead to crucial insights about electricity.

Merely labelling a promise body $-b$ is ambiguous, so we shall take it as given that nothing more than acquiring of information is implied by such a promise. A promise with body $-b$ is a specification of what behaviour will be "received" or not blocked by the agent.

3.9.4 Promises to use $U(\pi)$

The fact that an agent promises to receive, or not block a service, (e.g. collect the post from the postbox), does not imply that it will act on the content, or use it for any purpose. We therfore choose to emphasize the promise to use the promise of a service (e.g. open and read the letters from the postbox).

Although the use-promise is an outgrowth of a receive promise, it is debateable whether a use promise should be classified as a $-$ promise. However, a promise to use has a necessary pre-requisite to receive, so it is natural to either omit the $-$ part ($U(b)$), or shorten the notation to ($-U(b)$), (see table 3.4).

$$\begin{aligned} A_1 \xrightarrow{S} A_2 \quad \text{or} \quad A_1 \xrightarrow{+S} A_2 \\ A_2 \xrightarrow{U(S)} A_1 \quad \text{or} \quad A_2 \xrightarrow{-S} A_1 \end{aligned} \tag{3.35}$$

3.9.5 Coordination promises

In addition to these two fundamental types, it is possible to define other promise patterns with special meanings, such as coordination which we explain below.

- $C(\pi)$ a promise to coordinate with another agent
- $\pm\psi(\pi)$ a promise to reveal or heed knowledge about a promise π

Don't confuse the body describing the communicatio of a promise with the promise itself, as these are two different kinds of object.

3.9.6 Lies and deceptions

From the basic definition 2.12, we can begin to formalize the notion of a lie or deception. A lie, or deception is a promise made about something the agent knows it cannot accomplish, or that it has no intention of keeping.

- These are promises not respecting the default assumption, since default promises must be kept.
- Only the agent making the lie is capable of detecting it in general.
- A liar could try to avoid being caught by saying "I promise X if I can", but ....this is simply a superfluous promise, equivalent to an empty promise, since the conditional is left dangling. It is an evasion.

A set-formulaic way of representing a deception might be the following. A deception is asserting a promise that it does not intend to keep. i.e. an agent A_1 promises something b that is not necessarily kept to A_2, while telling a possibly third agent C that it will promise something that is both not b and which will be kept.

$$A_1 \xrightarrow{b \notin K} A_2 \tag{3.36}$$

$$A_2 \xrightarrow{b' \in (\neg b \cap K)} A_3 \tag{3.37}$$

where $A_3 \neq A_2$, i.e. A_2 does not hear the promise to not do b. e.g.

Symbol	Interpretation
$A \xrightarrow{+b} A'$	Promise from A to A' with body b
$A' \xrightarrow{-b} A$	Promise to accept b
$v_A(A \xrightarrow{b} A')$	The value of promise to A
$v_{A'}(A \xrightarrow{b} A')$	The value of promise to A'

Table 3.4. Summary of simplified promise notation

3.9.7 Compatible and incompatible promises

Exclusive promises are those which cannot physically be realized at the same time. This does not mean that incompatible promises cannot be made, it means that they are meaningless and could lead to problems for the agent.

Definition 24 (Incompatible promises $\#$) *When two promises originating from an agent are incompatible, they cannot be realized at the same time without contradiction. We write*

$$A_1 \xrightarrow{b_1} A_2 \# A_1 \xrightarrow{b_2} A_3 \tag{3.38}$$

If $A_2 = A_3$, we may omit the agents and write $b_1 \# b_2$.

When do we know that promises are incompatible? In general we cannot know this a priori, this is an assessment to be made by an agent. Different agents might make this judgement independently and arrive at different conclusions. Technologists will undoutedly be sceptical of this answer and look for a logical formula to decide when promises are incompatible, however this would require the existence of a common model which would violate relativity.

Example 4
- *Promises for a door to be kept open and kept closed at the same time.*
- *Promises to attend meetings in Oslo and Amsterdam at the same time.*

On the matter of trust, it would likely be unwise for an agent to trust another agent that made simultaneous, incompatible promises. Of course this is a policy decision for each individual agent to make.

Breaking a promise is not the same as not keeping a promise. It is an explicit contradiction. Again, confidence in an agent's promise-keeping ability is reduced when it makes contradictory promises.

Definition 25 (Promise conflict) *Two or more promises to an agent are in conflict if at least one promise is contradicted by another. We define a conflict as the promising of exclusive promises*

$$A_1 \xrightarrow{b_1} A_2,\ A_1 \xrightarrow{b_2} A_2 \text{ with } b_1 \# b_2 \tag{3.39}$$

Clearly promising b and $\neg b$ would be excluded.

3.10 Idempotence in repetition of promises

What if we repeat a promise to pay 100 euros to someone; have we kept our promise if we pay 100 or 200 euros?

Axiom 1 (Separate events have separate types) *Each new labelled promise that is fulfilled by the same type of action, must be labelled by a formally different type promise.*

Certain promise types are therefore related, or we can think of them as being parameterized. For example, the different payments above have different types, otherwise we lose track of total value. e.g. τ_1 is payment 1, τ_2 is payment 2.

Rule 1 (Idempotence of promises) *The promise combination operator is* idempotent*, so repeating the same promise does not change anything:*

$$\frac{A_1 \xrightarrow{b} A_2, A_1 \xrightarrow{b} A_2}{A_1 \xrightarrow{b} A_2}. \tag{3.40}$$

Or in simpler notation:

$$\pi(b) \cdot \pi(b) = \pi(b). \tag{3.41}$$

and if

$$\pi^n(b) \equiv \pi(b) \cdot \pi(b) \cdot \ldots \pi(b), \tag{3.42}$$

then by implication

$$\pi^n(b) \to \pi(b). \tag{3.43}$$

meaning that the repetition of the same promise n times is the same as a single promise. However, the belief in the outcome of the promise might be strengthened by the repetition of the promise, so we may write the notation:

$$\beta(\pi^n(b)) = \alpha(n)\beta(\pi(b)), \tag{3.44}$$

meaning that belief in the outcome is amplified in the repetition of the promise by some factor $\alpha(n) \geq 1$.

3.11 Promises to keep promises

Consider the implications of making promises about promises. Let us define

$$A_1 \xrightarrow{b(\pi)} A_2 \tag{3.45}$$

to mean a promise π to keep the conditions of the body $b(\pi)$, and let

$$\pi' : A_1 \xrightarrow{\pi} A_2 = A_1 \xrightarrow{A_1 \xrightarrow{b(\pi)} A_2} A_2, \tag{3.46}$$

mean a promise to keep the promise π. A promise to keep a promise might or might not be as strong as a direct promise. This does not affect the details of expected outcome, only belief in the likelihood of the outcome. Thus we cannot say that (3.45) and (3.46) are the same, but it is reasonable to say that they are equivalent up to a discounting factor $\delta \leq 1$ that affects the belief in the outcome. We may write the notation:

$$\beta(\pi') = \beta(\pi(\pi(b))) = \delta\beta(\pi(b)) \tag{3.47}$$

and if

$$\pi^{(n)}(b) \equiv \pi(\pi(\ldots \pi(b))), \tag{3.48}$$

then the possible outcomes are the same, so this is equilavent to a different promise with the same intent

$$\pi^{(n)}(b) \to \pi'(b). \tag{3.49}$$

however it is somewhat weaker. So we have

$$\beta(\pi^{(n)}(b)) = \delta(n)\beta(\pi(b)) \tag{3.50}$$

meaning that belief in a promise to keep a promise is weaker by a factor δ than a direct promise. This result may be contrasted with the idempotence or repeated invocation of promises in (3.41).

3.12 Exclusive or incompatible promises

Exclusive promises are those which cannot both be kept if the lifetimes of the promises overlap. This does not mean that incompatible promises cannot be made, it means that they are meaningless and could lead to problems for the agent. Consider a promise

$$A_1 \xrightarrow{b} A_2 \tag{3.51}$$

meaning a promise from A_1 of the body b to A_2. We can use a functional approach to extract information from the body, e.g. $\tau(b)$ means the promise type.

Definition 26 *Incompatible promises #. When two promises originating from an agent are incompatible, they cannot be realized physically at the same time. We write*

$$A_1 \xrightarrow{b_1} A_2 \# A_1 \xrightarrow{b_2} A_3 \tag{3.52}$$

If $A_1 = A_3$, we may omit the agents and write $b_1 \# b_2$.

Definition 27 (Broken promises) *There are two essential ways in which a promise can be broken:*

1. *The agent takes an action that is incompatible with the terms of the promise.*
2. *The agent makes another promise that is incompatible with the terms of the promise.*

$$\left.\begin{array}{l} A_1 \xrightarrow{b_1} A_2 \\ A_1 \xrightarrow{b_2} A_2 \end{array}\right\} b_1 \# b_2$$

Note that the second case includes the case where an agent consciously or unconsciously exceeds a boundary or limitation that is expressed in the promise body, e.g. exceeds a deadline in time, or steps over a boundary in space.

A promise broken is stronger than a promise not kept, as it implies an intended action by the agent. An intentional omission to act can also break a promise.

It is normally understood that a promise is broken by a later promse, but it is clearly unimportant that we exclude one point of simultaneity in time, at which one might simply say that two promises were inconsistent.

As we learn to combine promises from many agents, it can emerge from the logic that combinations of promises between multiple agents could lead to a promise being broken between two agents. This does not enlarge the definition, it only means that certain contellations of promises can lead the same agent to make and implicit or explicit promise that contradicts another that was freely given.

Rule 2 (Parallel promises must have different types) *Two parallel promises are compatible if*

$$\left.\begin{array}{l} A_1 \xrightarrow{b_1} A_2 \\ A_1 \xrightarrow{b_2} A_2 \end{array}\right\} \quad \tau(b_1) \neq \tau(b_2) \leftrightarrow \neg(b_1 \# b_2)$$

i.e. two compatible promises can be made in parallel if and only if they are not of the same type.

3.13 Promise conflicts

What if, in a component system, several component-agents make identical promises? In practice, promises might be similar, but they are rarely identical. For example, two brands of coffee promise the smoothest cup but taste very different, or

similar databases are MySQL and PostgreSQL, both ostensibly following the SQL standard, but, they are very different implementations with different focus. However, we will often perceive them to be the same 'for all intents and purposes'.

Example 5 *Think of two resistors in an electrical circuit. Resistors are marked with coloured bands which represent the promise that they make, e.g. 150 kilo-ohms, plus or minus 5 percent. Two resistors with the same markings might have difference resistances, hopefully within a ten percent range, if they keep their promises – the exact resistance of the components can only be found by using them.*

Suppose two components really do give identical promises. In that case, one is really an alias for the other. A user of the promise would not be able to distinguish the two promises from one another (see section 8.4.3). The real problem is one of imperfect information: a component sketch cannot truly know whether two components are identical without a 'try and see' approach. The components' promises might not be specific enough to advertise an intended difference. The real issue thus arises when indistinguishable promises lead to different behaviours. We would like to avoid this by avoiding approximation promises, where possible. In practice, this is hard to do, but this offers a guideline.

Example 6 *Imagine a shop that promised "We sell some kind of tea". A potential user of this promise, might search for all providers of 'some kind of tea', but would these be satisfactory? Often we need to be specific and the ability to search for alternatives is not helpful.*

Form section 3.9.3, it should be clear that promises predict two kinds of inconsistent promises, or conflicts: conficts of use $-$ and conficts of give $+$ promises. The duality property of promises in section 5.6 further suggests that, if a conflict can occur in a $+$ promise, then there must also be a corresponding conflict possible in a $-$ promise. Promise conflicts relate to the existence of exclusive promises, or contention over shared resources.

The problem is one of incomplete information. A component that wants a service from another component might not have sufficient knowledge of the other agent's promises to know whether can expect its promise to be kept. This can be traced back to the way designers of the components think in terms of obligations: function, please give me 'X' – in the manner of an obligation. To design components properly, we need to think in terms of vountary cooperation and sufficient information.

- **Conflict of give** A give-conflict is a conflict where an exclusive promise is made by an agent in a scope where several agents can see the promise. We can also think of this as a problem of uncoordinated usage.

  Suppose now two agents who want to rely on this promise (see fig. 3.2) both begin to rely on it without the consent of the giver. In our formal terms, this is an involuntary use of resources, which is an attack on the agent. The giver can promise either X or Y, but not both at the same time. Thus the fact that two users both want something at the same time causes contention for the giver.

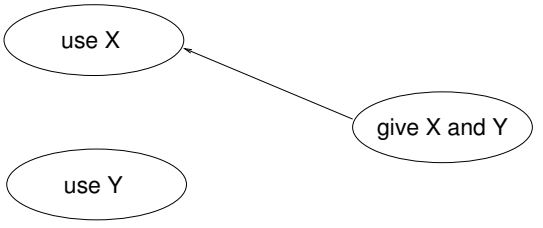


Fig. 3.2. Conflict of give

  This scenario happens in all service giving organizations, like call centers, web servers, banks, airline checking, supermarket checkouts, etc. Consider now two agents that both want access to the same promised resource X. We assume here that only one of them can use it at a time, i.e. there is only one server resource.

  This kind of confict can be avoided by the use of conditional promises. Instead of the giver promising X and Y without condition, it can promise X unless Y or Y unless X. Which of these two is then promised depends on the order in which the users arrive. If a user agent uses X first, then Y will be unavailable, and vice versa. In computer science, this is known as transaction locking, or 'mutex' (meaning mutual exclusion). It is a way to achieve serial access to a disputed common resource.

  A second workaround is to duplicate the resource itself In computer science, this is called 'threading' of the service. It is a multiple queue architecture, such as one sees at the checkout of a supermarket: rather than giving exclusive access to the cash register, one simply duplicates the service to be able to promise less contention.

- **Conflict of use**

The dual confict is one of inconsistent or uncoordinated giving. What if an agent relies on a promise from two or more providers in order to quench a single need? And what if the promises they keep are different? The result will be unpredictable and perhaps even superfluous; the actions of the giving agents will be uncoordinated.

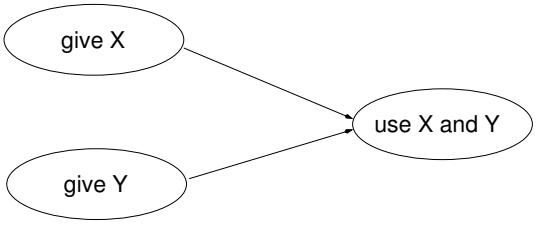


Fig. 3.3. Conflict of use

The resolution here is a policy for selection from the alternatives. Such a resolution is a part of the promise made by the user, thus this kind of confict is actually a symptom of incomplete specification. Part of a policy for handling muktiple sources can be a strateyg for fallback or redundancy.

We can state some simple design rules for agents that make promises to avoid exclusion, for those cases where one uses promises in a position of being able to design a system.

Rule 3 (Exclusive promises should be unique) *Exclusive promises should not be parameterizable so that they appear in multiple varieties. Each exclusive promise should have a unique promise body.*

Rule 4 (Advertise exclusivity) *Exclusive promises should be advertised as being exclusive, by making them conditional: e.g. I promise you exclusive access to X only if I can secure Y.*

Rule 5 (Limit scope of exclusive promises) *If one limits the scope of an exclusive promise to a single promisee, then there should be no contention for the exclusive resource.*

In designing systems, where promises are used to do bookkeeping of interactions, components making promises can often be redesigned to make promises non-exclusive. This can make systems more robust, but often at the cost of a more extensive system.

3.14 Promise refinement and overriding

Sometimes the idea of breaking a promise with a similar promise by is too strong. The concept of inheritance of default assumptions has become commonplace in computer programming. We could conceivably allow the presence of multiple promises as long as the order in which promises override one another is defined. In fact we only need to know which of the agents has "primacy" with respect to promises of a certain type $\tau(b)$.

Definition 28 (Promise refinement) *Let b and b^* be any two promises of type $t = \tau(b) = \tau(b^*)$. We can then say that these promises of the same type*

$$A_1 \xrightarrow{b^*} A_2$$
$$A_1 \xrightarrow{b} A_2 \ldots$$

are not broken iff $b^* \subseteq b$ and promise b^* has "primacy" (denoted by the asterisk). Only one promise of type $\tau(b)$ from any promising agent can have primacy.

In other words, a more restrictive promise does not contradict a more general promise as long as it has primacy.

Example 7 *The post office promises, by default, to deliver letters to the address printed on the front of the letter. However, this promise might be overridden by a promise to forward a letter to a change-of-address location.*

Example 8 *At a restaurant, menu item 5 might promise a dish as described in the menu specification. After indicating the he will only promise to accept item 5 if there are no carrots, menu item 5 promises to deliver the specification 5 (which includes carrots) without carrots. Thus the promise for carrots is overriden.*

Example 9 *Nearly all security access systems maintain lists called Access Control Lists that determine who can access certain resources. For instance:*

- *A supermarket promises to sell goods to customers, but only promises to sell beer to customers over 18 years old.*
- *An airport departure terminal promises to admit all persons with boarding cards. The airport lounge (which is part of the departure terminal) promises to admit only passengers with business class tickets or a gold card.*
- *A museum promises to admit anyone with a valid ticket, but the museum shop (which is part of the museum) promises to admit anyone.*

In many of these cases the issues are indicative of a deeper level of granularity in the promising agent. If we take the example of the museum, there is no need to override a promise if we recognize that the museum 'agent' can be decomposed into two different agents, one for the shop and one for the rest. Then each makes clear promises with no need to override. In the case of the supermarket, it could be conisdered a modelling error to view the supermarket as the promiser, with so much internal structure. A better approach might be to make each type of good promise availability to different customers, then there is no overlap or need to override. The main cause of overriding is a desire to avoid modelling internal details; however, then one questions the motivation.

This concept of multiple promises is somewhat undesirable, since it only complicates the logic of consistency. We believe it should only be used in exceptional circumstances when designing policies. However it will prove to be necessary in order to model the concept of overriding authority. One may signify this with a special notation $*$ on the primal promise body?

3.15 Assessment

How can an agent assess whether or not a promise has been kept? Is this even possible? In the general case, this could involve sophisticated ideas, like *knowing*, which is a complex subject.

Eventually we would aim to build a theory of observation or measurement, allowing us to embed promise theory into more familiar theories of the natural world, with all of the practical benefits of empiricism and engineering. We return to that issue in chapter 8. However, in the absence of that theory of measurement, the simplest option is to treat this issue as a semantic primitive. Thus, for now, we make assessment a black box operation.

3.15.1 Kept and not-kept

Assuming that the promise in fact has meaningful content, one could move to the issue of assessing whether or not it has been kept. However, at this stage, this issue is too big to confront in a systematic and comprehensive way, so we shall temporarily side-step the issue pending a longer discussion in chapter 8, by making this issue a semantic primitive, i.e. we do not define this in terms of other notions.

Definition 29 (Kept promises) *In a subset of histories, a promise made in that history has been 'kept'. In all others it hasn't.*

We now want a notation for saying that a promise has been kept. We can refer to the promise as a triplet $\langle A_1, b, A_2\rangle$ or as $A_1 \overset{b}{\to} A_2$, so we write either $K(A_1, b, A_2)$ or

$$K\left(A_1 \overset{b}{\to} A_2\right). \tag{3.53}$$

To include the history, we want to say that H 'satisfies' the keeping of the promise:

$$\begin{aligned} & H \models K(A_1, b, A_2) \\ \text{or } & H \models K\left(A_1 \overset{b}{\to} A_2\right). \end{aligned} \tag{3.54}$$

Mark finds this notation unfamiliar and would tend to write something like

$$K_H\left(A_1 \overset{b}{\to} A_2\right), \tag{3.55}$$

meaning $K_H(\cdot)$ is the 'kept within history H' function.

3.15.2 Minimal histories

Let $\pi = \langle A_1, b, A_2\rangle$ be a promise.

Definition 30 (Positive promise) *A promise π is called positive iff, for all H, H'*

$$H' \geq H \Rightarrow \big(H \models K(\pi) \Rightarrow H' \models K(\pi)\big) \tag{3.56}$$

i.e. if a promise to make something happen has been kept within a certain history H, then it has still been kept within a larger history H'.

The keeping of a positive promise *persists* with time. There is therefore a notion of a minimal history in which a promise will be kept.

A promise not to do something is a negative promise.

Definition 31 (Negative promise) *A promise π is called negative iff, for all H, H'*

$$H' \geq H \Rightarrow \big(H' \models K(\neg\pi) \Rightarrow H \models K(\neg\pi)\big) \tag{3.57}$$

i.e. if a promise to make something not happen has not been realized by a certain elapsed time, then it was not realized before that time either.

Notes

[6]We have taken a leap in making this definition of a promise, which might be considered too much to swallow by our readers. Appendix A explains in more detail a simple line of reasoning for making the choices implicit in this definition.

[7]Notice that the scope of the promise is not the scope of the behaviour, but the extent of its publicity. The body of the promise describes the extent to which it affects the world.

[8]The classical theories of relavity by Galileo and Einstein deal with all the same issues as we must deal with here for continuum models. A model of agents is a discrete network so the notions of speed of information are not defined in the same way.

[9]Think of knowledge as knowing something as you know a friend.

[10]Note that in Quantum Mechanics, there are no forces, only expected outcomes, so this would not be the first time something so audacious had been proposed.

[11]Promises bear a passing resemblance to the theory of capabilities in ref. [Sny81]

4
Combining Promises into Patterns

The usefulness of promise concepts lies largely in their atomic properties. However, it is the way we build larger structures by combining promises into *systemic designs*, which accounts for the usefulness of a theory in a man-machine environment. This chapter describes a few such patterns.

4.1 Promise structures

We begin with the notion of grouping related promises under a collective name. Since promises are always autonomous parts, grouping has a purely semantic significance. It does not alter the behaviour of any part of a potential 'system'.

The simplest structure on top of multiple promises is to collect promises into a kind of logical container with a name. We call this a promise *bundle*. Promise bundles form the basis for understanding system *components* and the functional *roles* they play. They can be defined as absolute quantities, or as parameterized patterns to be re-used as templates for expressing variation.

4.1.1 Bundles of promises

A basic promise bundle is any collection of promises between any set of agents that we choose to group into an entity.

Definition 32 (Promise bundle) *Let $S, R \subseteq A$ be arbitrary sets of agents, possibly overlapping. A promise bundle is any collection of promises made from agents in S* onto *agents in R. We denote this*

$$S \stackrel{B}{\Longrightarrow} R \equiv \{s_1 \stackrel{b_1}{\longrightarrow} r_1, \ldots, s_1 \stackrel{b_n}{\longrightarrow} r_n\}. \tag{4.1}$$

B is the name of the bundle, and reprsents the collective content of the member promises.

For example, we might choose to collect all our promises about driving a car into a driving bundle, and all of our promises about healthy eating into a diet bundle.

Bundles are logical containers, and they can overlap. The same promise can be part of multiple bundles, without any difficulty. The main purpose of bundles is to offer a way of assigning names to simple functional patterns.

Some bundles might have special properties, e.g. bundles that map a number of parallel promises between the same set of sender and receiver agents. We might call these bundles homogeneous.

Definition 33 (Homogeneous promise bundle) *Let $S, R \subseteq A$ be arbitrary subsets of equal dimension d, from the set of agents. A promise bundle is any collection of promises made (one-to-one) from agents in S* onto *agents in R. We denote this*

$$S \stackrel{B}{\Longrightarrow} R \equiv s_1 \stackrel{b_1}{\longrightarrow} r_1, \ldots s_d, \stackrel{b_d}{\longrightarrow} r_d. \tag{4.2}$$

where Π denotes a bundle of promises and B is the collection of promise bodies $b_1, b_2, \ldots, b_d$.

Promise bundles will be important when defining the parallel composition of promises into patterns, especially for the purpose of discussing system organization and componentization.

4.1.2 Parameterized bundles

A parameterized bundle is a pattern like a template, made up of conditional promises, in which some of the conditions to be fulfilled are promises to the use or accept information from external agents. In a computational sense, parameterized patterns are like functions with arguments. We plug in certain values in order to re-use a generic pattern. The value returned by the function could be interpreted as the collective state of promises kept and not kept.

Definition 34 (Parameterized promise bundle) *Let B be the collection of promises with bodies $b_1, b_2, \ldots, b_d$ belonging to a promise bundle. The bundle is said to be parameterized if at least one of the bodies b_i represents a promise to use information from an external agent, which is subsequently used to specify the choices made in the other promises.*

Example 10 *A website might promise to deliver online goods, but you have to promise to tell them which ones. The bundle of promises the website would keep might be basically the same for any choice, but would still depend on the exact choice input by the customer. Thus the bundle would include a promise to use information from the customer, before dispatching anything.*

Example 11 *In the computer language CFEngine, a parameterize bundle is written as in the example below. This bundle consists, for illustration only, of a single promise whose body consists of statements of the form $attribute \Rightarrow value$. The parameter values are subsituted by writing* `$(name)`*:*

```
bundle myname(parameter1, parameter2)
{
files:                                          # The promise type

    "$(parameter1)"                             #  The promiser

         create => "true",                      # The body
      edit_line => "$(parameter2)";             #
}
```

The bundle has a name 'myname' and two parameters corresponding to use-promises. When this bundle is promised one promises values for these parameters at the same time:

```
  "promiser"
    usebundle => myname("file1", "content of file");
```

Parameterized bundles are an important pattern for the practical use of promises, as they allow us to make generic 'forms' that compress the total amount of work into smaller, reproducible promise templates. Many standard contracts have this property, such as application forms, etc.

4.2 Agent roles as promise patterns

Once we have the concept of bundles, we can imagine that these play different roles in a system of promises. We may, in other words, use bundles to decompose sets of promises into possibly overlapping categories, in order to be able

to fit our understanding of the system according to any particular mental model. This is a realistic approach to understanding function in a network of agents[12]. The concept of roles from promises was first discussed in [BFa, BFb, BF06]. In this section we shall try to formalize the notion of a role. This will be key to understanding the composition / decomposition of systems.

4.2.1 Formal definitions of kinds of role

We can imagine roles from coming from a number of different interpretations of what identifies a subset of promises.

- Groups of agents that send S and receive R promises (i.e. promisers and promisees).
- Grouping agents that give the same $+$ promise (i.e. functional similarity) – role by association.
- Grouping promisees that are made the same promise (i.e. recipient similarity) – role by appointment.
- Grouping agents that use the same promise (i.e. functional dependency).
- Grouping agents that work together so that a representative can make a promise like a single agent (cooperation).

Roles may thus be associated both with *types* of promises τ, compositions of promises and agents, and the topology of the promise graph itself. We shall meet roles in several other situations, including in the definition of components and organizations.

In this respect, we can associate a collection of agents with a role, given that they make (or use) the same set of promise types. In other words, we are guided by the typed structure of the graph alone.

Definition 35 (Roles by association) *Agents that* make *the same promise may be grouped by their associated function. Let $\mathcal{A}$ be any set of autonomous agents, divided into senders S and receivers R. The family of sender subsets in a promise graph, which take part in promises with body types $t(b)$, define b-roles by association. Note that the promise made might be a give or receive promise (i.e. of type $\pm$).*

Definition 36 (Roles by appointment) *Agents that are* made *the same promise may be grouped by their associated function, regardless of whether they themselves promise to use the promise given. Let $\mathcal{A}$ be any set of autonomous agents. The family of receiver subsets $R \subseteq \mathcal{A}$ in a promise graph, which take part in promises with body types $t(b)$, define b-roles by appointment.*

Note that promise roles are identified empirically, not pre-decided by design.

Example 12 *The set of all agent nodes $W \subseteq \mathcal{A}$ that promises to provide web service to any agent $R \in \mathcal{A}$:*

$$W \xrightarrow{\text{web}} (R \in \mathcal{A}) \tag{4.3}$$

plays a role, which we can call 'web server'. This is called a role by association, since the members of W are not connected in any other way than they make the same promise. The set of nodes $C \subseteq \mathcal{A}$ that is promised web service by any agent:

$$(S \in \mathcal{A}) \xrightarrow{\text{web}} C \tag{4.4}$$

plays a role, which we can call 'web client'. This is called a role by appointment, since the members of C are pointed to by a promise of the same type.

Example 13 *In an electronic circuit, resistors and capacitors play roles by association. They are agents that make promises to resist current and store charge, respectively, within specified limits. Resistors with different values might play different roles depending on how specifically the promises are stated.*

Example 14 *In the promise-based software CFengine, the concept of 'design sketches' is used to bundle certain promises together that play a specific design role. CFEngine's sketches play roles that have interesting composition properties. For a detailed discussion, see section 13.7.*

Example 15 *Membership clubs, leaders and pop stars are all examples of roles by appointment. Agents (people) promise to follow them and subscribe to their activities, whether they are aware of this or not.*

The example of an electronic circuit is a useful one because we can all relate to something functional that is built from smaller parts. On the one hand, we can see the roles played by agents (components) within the circuit at a low level, but then at a higher level there is also the role of the radio or the television. The radio function is a cooperative phenomenon that comes from the promises being kept together in a coordinated way.

Definition 37 (Roles by cooperation) *Let $\mathcal{A}$ be any set of autonomous agents. A subset of agents C that are collectively involved in providing a promise with body types $t(b)$, define b-roles by cooperation. In order to cooperate, all agents in the group supporting the role promise $C(b)$, i.e. to cooperate in the performance of b, i.e. C is a cooperative group if and only if each agent participates in a mutual agreement to coordinate with every other agent in the promise:*

$$C \overset{b}{\to} a_i \tag{4.5}$$

for some node or set of nodes a_i. Every agent in the group has a promise

$$C \overset{C(b)}{\to} C \tag{4.6}$$

for the given b. (See fig. 4.2)

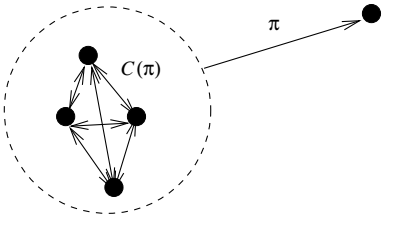


Fig. 4.1. A cooperative role is defined when a group, formed through mutual cooperation, agrees to offer the same promise to another node, or set of nodes.

4.2.2 Graphical interpretation of roles.

Because promises form a graph, or behavioural network, there is a natural topological understanding of what it means for an agent to play a specific role in a system. Agents that make the same patterns of promises, may be said to play the same roles.

In technical terms, we are guided by the topology of the typed graph alone[13].

- An *appointed role* arises when an agent or group is being pointed to by a set of identical promise arrows (role-sink). In a sense, the agents are effectively voted for (like the candidates in an election).
- Roles by *association* are played by uncoordinated agents that happen to make the same promises, such as the voters in an election.

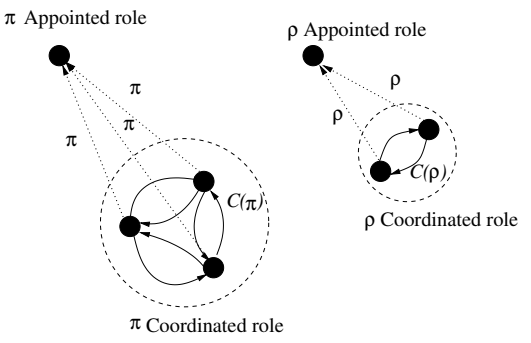


Fig. 4.2. Typed roles distinguish between the type of promise used to define the structure. Natural groupings result from the autonomous behaviour.

- A *cooperative role* arises when a group of agents is united by making a coordination/subordination promise to one or more other nodes. They take on the role of a 'community' or 'organization' of dependables.

4.3 Collective behaviour

To make predictions about a system, i.e. discern its probable behaviour we need to know the promises made by its component parts. The extent to which a system behaves in a collective manner or as independent agents is about the existence of promises to direct their activities in a coordinated manner. Two kinds of collective behaviour receive special interest:

Definition 38 (Coordinated behaviour) *Agents are said to exhibit coordinated behaviour if they make promises to one another.*

Coordination need not imply a correlation between agents, only that they have made their intentions known to one another. For example, the two promises 'I will do the shopping' and 'I will watch the kids' are coordinated promises that are uncorrelated.

Cooperation implies more than coordination. It is a form of programming towards a single collective goal.

Definition 39 (Cooperative behaviour) *Two agents are said to engage in cooperative behaviour if each agent makes promises to other agents (possibly each other) that contribute to a common outcome.*

Cooperative behaviour includes, of course, 'non-cooperative' behaviour. Agents that work together towards a common goal will therefore exhibit coordinated cooperative behaviour.

It is not always a simple matter for an observer to distinguish coordinated behaviour from cooperative behaviour, if the observer does not have access to the underlying promises. We shall argue that this is the origin of so-called 'emergent behaviour' in chapter 7. The ability to distinguish behaviour through observation will eventually play an important role understanding systems with real and apparent promises.

4.4 Goals, or special intentions

When agents work together, one often speaks about the goal of their activities, implying an alignment of purpose. The term goal is not significantly different from an 'intention', indeed there is no reason to inflate the terminology with a new concept. Just as commitments are merely promises that an agent has singled out as being deserving of a special name due to a sense of priority, so goals are special intentions that may or may not have been made public. When agents work together, they might or might not share the same goals, even if they make the same promises.

Definition 40 (Goal) *A goal is an intention with special significance to the agent that maintains is.*

The idea of a goal brings back the issue of value judgement, or how an agent singles out particular promises as being special. We return to this issue more fully in chapter 14. In short, if a goal is met, i.e. the intention was realised with a positive outcome, then it can be said to yield positive value to an agent. There is thus a simple numerical or economic approach to the identification of special intentions.

4.5 Examples of μ-patterns

4.5.1 Network load Balancing

A load sharing or load balancing system is an example of what it means to build a collaborative computing structure. It has features in common with the firewall example we mentioned earlier, in that it involves end-to-end servive through intermediary agents. Such an archtitecture is usually designed as shown in fig. 4.3. Let us now aim for a system that offers a service by collaborative effort. What kinds of promises would lead to this behaviour?

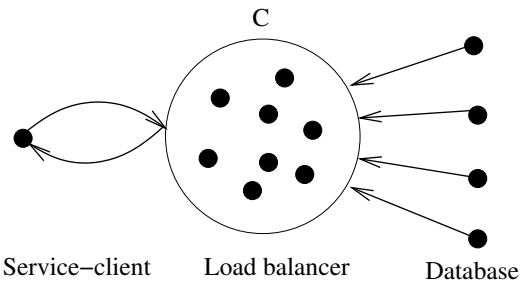


Fig. 4.3. A schematic promise graph of a typical load balancing configuration. The circular region represents a fully connected graph of collaborative agreements, making all the indexing agents behave equivalently towards a service-client. Their mutual agreements define their role in this collaboration. These in turn are promised individually service by four database servers.

Consider the example in fig 4.4. Let C be the client, S_i be a set of servers for index i, and LB be the load balancer. Let us ignore the trivial complication of how LB selects a particular S_i.

- Case (a): this is a relay architecture such as would be used in Network Address Translation:

$$LB \xrightarrow{-\text{request}} C \tag{4.7}$$
$$LB \xrightarrow{\text{serv\_req}} S_i \tag{4.8}$$
$$S_i \xrightarrow{-\text{serv\_req}} LB \tag{4.9}$$
$$S_i \xrightarrow{\text{serv\_reply}} LB \tag{4.10}$$
$$LB \xrightarrow{\text{serv\_req/serv\_reply}} C \tag{4.11}$$
$$LB \xrightarrow{-\text{serv\_reply}} C \tag{4.12}$$

The load balancer node promise to receive requests from a client (perhaps by arranging access control etc) and to forward service requests to a typical server S_i. This server promises to receive the request and return a service reply. The last to promises constitute a promise to relay the reply back to the client.

- Case (b): this is a dispatcher architecture with direct reply.

$$LB \xrightarrow{-\text{request}} C \tag{4.13}$$
$$LB \xrightarrow{\text{serv\_req}} S_i \tag{4.14}$$
$$S_i \xrightarrow{-\text{serv\_req}} LB \tag{4.15}$$
$$S_i \xrightarrow{\text{serv\_reply}} C \tag{4.16}$$

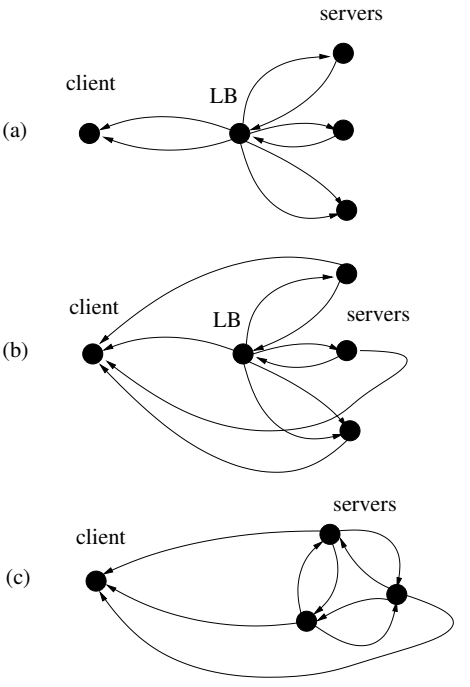


Fig. 4.4. Load balancing with central and peer to peer promises. Cases (b) and (c) give equivalent promises to the client, only with a different role binding model. Remember that the arrows do not represent information flow. The promises reveal the topology of the system architecture.

In this case, the relay promises are not needed.

- Case (c): there is no dispatcher, the servers have simply agreed to collaborate on replying the client, each being an equal partner in this. Thus the client selects to server of choice (e.g. using DNS round robin, or a private list).

$$S_i \xrightarrow{-\text{request}} C \tag{4.17}$$

$$S_i \xrightarrow{\text{C(serv\_req)}} S_i \tag{4.18}$$

$$S_i \xrightarrow{\text{serv\_reply}} C \tag{4.19}$$

In this case, there is no single point of contact. The collaborative group formed by the cooperation promises are optional, and show how the servers might agree on the details of the service being provided. Without such agreements, the client

would not be able to verify that the service from each of the servers was identical. Clearly, we can embellish the system to allow peer to peer server-side balancing also and the three-tier architecture with a common data source. We leave these as an exercise to the reader.

We mention, in passing, another example of this kind of cooperation, namely routing protocols. Each network router makes a priori no promises to any other device to forward packets. It might forward packets or not. For instance, if it has only a static routing table, it makes no promises to forward packets but it does so. By promising to partake in a routing protocol collaboration like RIP or OSPF[Hui00], on the other hand, the router *subordinates* itself to information from neighbouring routers just as they do symmetrically. Each router then promises to relay packets from its neighbours according to the algorithm agrees upon in the collaborative role.

4.5.2 Network DNS configuration

Let's see how promise theory helps us to understand a well known service that is based on voluntary cooperation: the Domain Name Service (DNS). We shall suppress details from the promise bodies. If one thinks in traditional service oriented terms, it is tempting to think that it is trivial to model this service by a single promise from a server to each client. The promise made by the DNS server to reply to requests is certainly important (fig 4.5):

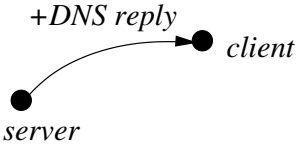


Fig. 4.5. First model of DNS.

The client, after all, makes no promise to send requests to the server. That is not a predictable matter to be encoded by a promise. The server, on the other hand, makes a general promise (usually to all agents with IP addresses) to provide information. This, in fact, involves two promises (fig 4.6).
i.e. a promise to receive the requests from clients and a promise to provide data in reply. Again, this is an entirely one-sided affair. This might make us suspicious: what do these promises correspond to? Does the client have no responsibilities in this matter? Indeed, readers might be worried that the client should have to 'send something' in order to get a reply? This is true enough, but

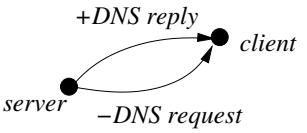


Fig. 4.6. Second model of DNS.

promises do not represent anything that is sent; they are not a communication protocol. In promise theory we are thinking above the level of communication exchanges. A promise is not a message. The promise made by the server to reply is not the reply itself, it goes beyond any single reply. It is a responsibility to be fulfilled.

Turning high level promises into low level implementation details can be part of a discipline that helps us to understand systems from the top down. A problem ontology can be useful here. Conversely, one can see how to replace a high level promise with many more specific low level promises, or build high level ones from low level ones bottom-up. At the management level, we are interested in the promises, but at the technical level we need to know how to implement them. In this case, the answer is quite easy: a promise by the server to use data from the client can be implemented by making sure that there is a server process with an open port 53 and appropriate firewall access rules to admit requests from the client receiving the promise (usually this is everyone, but it need not be). The promise to provide answers is also covered by running the daemon and having updated zone files containing the data.

Now let us look at the client more realistically. In name resolution is a client program (e.g. a web browser) needs to look up data in the name service. This lookup is not performed directly; there is an interloper: a local name resolver that promises local name resolution (fig 4.7).

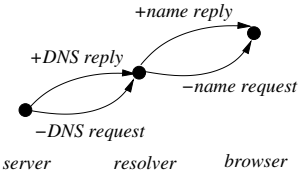


Fig. 4.7. Third model of DNS.

The web browser that makes no promises about its behaviour, although it avails itself of the resolution service at random times, but the local resolver

promises to give some answer to the web browser and to accept its requests. In all modern operating systems, this promise is guaranteed internally by the kernel architecture. These promises already indicate a number of things that can go wrong with the DNS system, if the necessary conditions are not in place to keep them. Every promise is a potential cause of failure.

The example is not yet complete however. There is, so far, no automatic connection in the promise graph above between the promise of name resolution and a promise of DNS resolution. The resolver can fulfill its promise to the client by saying "I don't know" to every request. It agreed to heed the request, it agreed to reply, but it did not agree to forward the request to an authoritative database in order to relay its response. We need two more promises (fig 4.8):

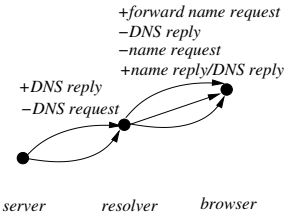


Fig. 4.8. Fourth model of DNS.

i.e. a promise by the resolver to the client to forward the request to a known DNS server, and a promise from the resolver to the client to use the reply it gets from that source. This promise is ususally implemented in Unix, for instance, by configuring the resolver subsystem in an `nsswitch.conf` file or equivalent. The promise to return a name is now conditional on a promise of DNS reply, and thus it is an empty promise unless it can also promise that it will receive such a promise.

Notice that, as we add the promises, the logic of the system reveals itself at a high level, without prejudice about architectures or other considerations (e.g. there is no reason to compare this to object oriented design or Unified Modelling Language[ABH]). This helps us to understand and even construct systems based on *specification primitives* (promises). The promises are directed towards the party that is interested in them. If we were thinking communications bindings we might think that the promise to use or receive the DNS reply would go to the agent that sent it (the DNS server), but the server doesn't care whether we use the reply or not. It has discharged its responsibilities by simply replying. What the promise says is that the resolver should not merely ignore the reply but use it in formulating a reply to the client. The additional promise is required

to complete the logic of promise combination.

There is more to DNS than these promises, of course. We can go still further and start to look at the redundancy of master and slave servers, and how they promise to give the same answers by coordinating their information. The master has to promise its data to the slave and the slave has to promise to use the data, as well as provide the same kind of service as the master. All this requires the coordination of even more promises. When we add all of these things we end up with this picture (fig 4.9).

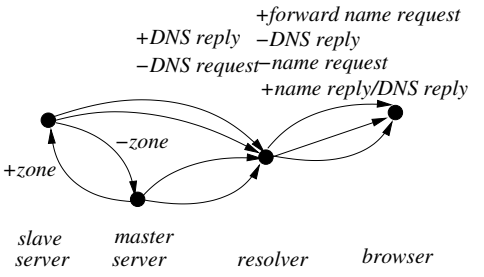


Fig. 4.9. Fiftth model of DNS.

Now we see the many things that can (and often do) go wrong with DNS, and the redundancy in the graph that makes it more robust to such failures. Due to the primitive nature of each promise and the autonomy of each component, the promise graph has the essential features of a fault network[HR94] and can be readily transformed into one. The promises thus allow us to see where things can go wrong at the highest level, without getting embroiled in implementation details. We have also applied the technique to software engineering[ABH].

The point, in this example, is not that we could not have figured all this out without the promises, but that we can start from the simplest drawings, in a constructive manner, without knowing the details of BIND or `gethostbyname()` or different operating systems etc.; we have predicted all of the potential problems by sketching out the main agents and by considering what kinds of promises are needed to get them working together. In a logic approach to policy confict analysis, one must first have the entire system in place to test whether it fulfils certain properties.

Notes

[12]Notice that the decomposition is not necessarily a hierarchy (or particular spanning tree) – it can be any identification of subset patches that we like. Thus we are not locked into an authoritative taxonomy of roles.

[13]Roles have been defined many times in computer science. Bishop defines them to be groups[Bis02] that tie membership to function. When a subject assumes a role, he or she is given certain skills or rights within the system. This delegation of rights and capabilities is performed by some central monitor, usually a system kernel. A role, according to Canright and Engø[CEM04] on the other hand, is a type of node that defines the topology of the total graph. A role can be source, sink, bridge, a member or a region, or a dangler (a dead end that is both a source and a sink).

5
Reasoning About μ-Promises

To secure a clear and consistent model of intent we must use formal methods, but what formalism do we have to express the necessary issues? Previous attempts to discuss the consistency of distributed policy have achieved varying degrees of success, but have ultimately fallen short of being useful tools except in rather limited arenas. For example, modal logics require one to formulate hypotheses that can be checked as true/false propositions. This is not the way policy-makers work.

MANY WORLDS??

THE GRAPHICAL NATURE OF PROMISES

5.1 Promise consistency and scope

How can agent promises be inconsistent? Is a conflict of intention enough to speak of inconsistenct? In fact it is not, because we must deal with the problem of scope. Two agents might have conflicting intentions with respect to some imaginary higher purpose, but have no knowledge of each other or this imaginary purpose.

For example: A intends to have his son take over his trading business when he retires. A's son intends to go to college and become a scientist. A's wife intends to have him marry the daughter of a friend and work the land to support them. Are any of these intentions consistent or inconsistent? What imaginary goals does one measure such consistency against?

Promises may only be made by a specific agent, so a more careful question is to ask how can an agent be inconsistent in making its promises? At the extreme pole where every agent is independent and sees only see its own world, there would be no need to speak of inconsistency: unless agents have promised to

behave in a similar fashion, they will do as they please. This is the property of a local theory.

Lemma 1 (Locality or promises of the first kind) *An agent cannot make an elementary promise (of the first kind) about any agent other than itself, or its own behaviour.*

A contradiction can still occur if a single agent promises two contradictory things. We call such a case incompatible or even broken promises. When all information is promised by (and hence localized in) a single agent, there is no need to go beyond that agent to look for inconsistencies. If a specific agent promises to marry and not to marry, then there is an obvious conflict. the locality means we only need to look in one place for this.

Definition 41 (Inconsistent promises) *Let b_1 and b_2 be promise bodies of the same type, i.e. about the same issue. A promise of b_1 from agent A_1 to agent A_2 is said to be inconsistent if there exists another promise from A_1 to A_2, of b_2, in which $b_1 \neq b_2$, i.e. if $b_i = (\tau, \chi_i)$, then $\chi_1 \neq \chi_2$.*

In the worst case, one could consider promising inconsistent promises to be a case of breaking both promises. This definition is very simple, and becomes most powerful when one identifies promise *types*. It implies that an agent can only break its own promises: if an agent promises two different things, it has broken both of its promises.

5.2 Indirection - a problem for deceptions

We have said earlier that the promise body should not normally contain references to agents. When agents' names are absent from the promise body, the logic is very simple. This is a desirable property.

But there are promises that cannot be expressed like this. Some complicated promises can be made (like the transference of responsibility example in [Bur05]). However there are cases where we cannot avoid referring to third parties. e.g.

- I promise I love you.
- I promise I do not love Mary.
- I promise Mary I do not love you.

Or "I promise to you not to promise X to Y", etc. These levels of complexity must be built up step by step.

This problem cannot be avoided if we keep to the rule that the promise body may not refer to other agents, since one could reframe the promises to avoid mentioning by name and providing the data about "whom I love" as a service.

This matter can be resolved as incompatible promises. We need some kind of taxonomy of promises that are related and incompatible. Promises need to be classified into some kind of taxonomy/ontology in order to know when certain promises are incompatible. e.g.

"I promise I love you"

"I promise Mary I don't love you"

These are the same promise type. There is no impediment to promising this, but the result is, of course, necessarily a lie.

This is easily seen be recognizing that there is only one type of promise made to two different promisees. The second promise thus breaks the first, by being incompatible with the first.

5.2.1 Di-graphs

In our definition of lies, we made an unwarranted assumption. There is no probelm with the following:

$$\begin{array}{c} A_1 \xrightarrow{\pi} A_2 \\ A_1 \xrightarrow{\neg\pi} A_3 \end{array} \tag{5.1}$$

A_1 can promise constadictory things to different agents without making any contradiction, provided the promise body does not refer to agents.

5.2.2 Tri-graphs

This is a lie to either A_2 or A_3, i.e. the assertion of two or more inconsistent promises to a promisee, about the behaviour of promising agent towards either the promisee or towards a third-party.

$$\begin{array}{c} A_1 \xrightarrow{\text{Love you}} A_2 \\ A_1 \xrightarrow{\text{Don't love A}_2} A_3 \end{array} \tag{5.2}$$

5.3 Promise analysis

Logic is a way of analysing the consistency of assumptions. It is based on the truth or falsity of collections of propositions $p_1, p_2, \ldots$. One must formulate

these propositions in advance and then use a set of assumptions to determine their status. The advantage of logic is that is admits the concept of a proof.

Is there a logic that is suitable for analyzing promises? Modal logic has been considered as one possibility, and some authors have made progress in using modal logics in restricted models[Ort98, GMP92]. However, there are basic problems with modal logics that limit their usefulness[LS97].

More pragmatically, logic alone does not usually get us far in engineering. We do not usually want to say things like "it is true that $1 + 1 = 2$"? Rather we want a system, giving true answers, which allows us to compute the value of $1 + 1$, because we do not know it in advance. Ultimately we would like such a calculational framework for combining the effects of multiple promises. Nevertheless, let us set aside such practical considerations for now, and consider the limitations of modal logical formalism in the presence of autonomy.

5.3.1 Modal Logic and Kripke Semantics

Why have formalisms for finding inconsistent policies proven to be so difficult? A clue to what is going wrong lies in the many worlds interpretation of the modal logics[Che80]. In the modal logics, one makes propositions p, q etc., which are either true or false, under certain interpretations. One then introduces modal operators that ascribe certain properties to those propositions, and one seeks a consistent language of such strings.

Modal operators are written in a variety of notations, most often with $\Box$ or $\diamond$. Thus one can say $\Box\, p$, meaning "it is necessary that p be true", and variations on this theme:

$\Box\, p$	$\diamond p = \neg\Box\, \neg p$
It is necessary that p	It is possible that p
It is obligatory that p	It is allowed that p
It is always true that p	It sometimes true that p

A system in which one classifies propositions into "obligatory", "allowed" and "forbidden" could easily seem to be a way to codify policy, and this notion has been explored[Ort98, GMP92, PS97, LS97].

Well known difficulties in interpreting modal logics are dealt with using Kripke semantics[Kri63]. Kripke introduced a 'validity function' $v(p, w) \in \{T, F\}$, in which a proposition p is classified as being either true of false in a specific 'world' w. Worlds are usually collections of observers or agents in a system.

Consider the formulation of a logic of promises, starting with the idea of a 'promise' operator.

- $\Box\, p =$ it is promised that p be true.
- $\diamond p = \neg\Box\, \neg p =$ it is unspecified whether p is true.
- $\Box\, \neg p =$ it is promised that p will not be true.

and a validity function $v(\cdot, \cdot)$.

5.3.2 Further problems with modal logic

Some modal logics begin the with assumption of idempotence of the core operators. For example, one assumes that $\Box\ \rightarrow \Box\, p \Box\, p$. Many justifications for this have been attempted; indeed, the left hand side can be phrased is several ways in English:

- It is necessary that p be necessary.
- p is necessarily a necessity.
- It is required that p be necessary.
- p must be necessary.
- p must be a requirement.

In each of these cases, the truth of this relation (the necessity of necessity) seems to be simply false in a world of autonomous promises, thus we reject this form of reasoning, autonomously and voluntarily (see section 5.4).

5.3.3 Single promises

A promise is usually shared between a sender and a recipient. It is not a property of agents, as in usual modal logics, but of a pair of agents. However, a promise has implications only directly on the behaviour of the promiser, not the promisee.

Consider the example of the Service Level Agreement, above, and let p mean "Will provide data in less than 10ms". How shall we express the idea that a node A_1 promises a node A_2 this proposition? Consider the following statement:

$$\Box\, p, \quad v(p, A_1) = T. \tag{5.3}$$

This means that it is true that p is promised at node A_1, i.e. node 1 promises to provide data in less than 10ms – but to whom? Clearly, we must also provide

a recipient. Suppose, we try to include the recipient in the same world as the sender? i.e.

$$\Box\, p, \quad v(p, \{A_1, A_2\}) = T. \tag{5.4}$$

However, this means that both nodes A_1 and A_2 promise to deliver data in less than 10ms. This is not what we need; a recipient is still unspecified. Clearly what we want is to define promises on a different set of worlds: the set of possible links or *edges* between nodes. There are $N(N-1)$ such directed links. Thus, we may write:

$$\Box\, p, \quad v(p, A_1 \to A_2) = T. \tag{5.5}$$

This is now a unique one-way assertion about a promise from one agent to another. A promise becomes a tuple $\langle \tau, p, \ell \rangle$, where τ is a theme or promise-type (e.g. Web service), p is a proposition (e.g.deliver data in less than 10ms) about how behaviour is to be constrained, and ℓ is a link or edge over which the promise is to be kept. All policies can be written this way, by inventing fictitious services. Also, since every autonomous promise will have this form, the modal/semantic content is trivial and a simplified notation could be used.

5.3.4 Regional or collective promises from Kripke semantics?

Kripke structures suggest ways of defining regions over which promises might be consistently defined, and hence a way of making uniform policies. For example, a way of unifying two agents A_1, A_2 with a common policy, would be for them both to make the same promise to a third party A_3:

$$\Box\, p, \quad v(p, \{A_1 \to A_3, A_2 \to A_3\}) = T. \tag{5.6}$$

However, there is a fundamental flaw in this thinking. The existence of such a function that unifies links, originating from more than a single agent-node, is contrary to the fundamental assumption of autonomy. There is no authority in this picture that has the ability to assert this uniformity of policy. Thus, while it might occur by fortuitous coincidence that p is true over a collection of links, we are not permitted to *specify* it or demand it. Each source-node has to make up its own mind. The logic verifies, but it is not a tool for understanding construction.

What is required is a rule-based construction that allows independent agents to come together and form structures that span several nodes, by *voluntary cooperation*. Such an agreement has to be made between every pair of nodes involved in the cooperative structure. We summarize this with the following:

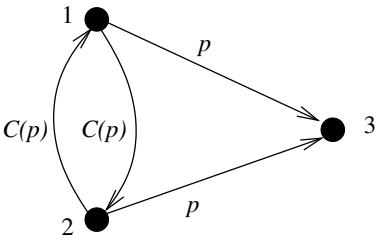


Fig. 5.1. (Left) Cooperation and the use of third parties to measure the equivalence of agent-nodes in a region. Agents form groups and roles by agreeing to cooperate about policy. (Right) This is how the overlapping file-in-directory rule problem appears in terms of promises to an external agent. An explicit broken promise is asserted by file, in spite of agreements to form a cooperative structure.

Assumption 1 (Cooperative promise rule) *For two agents to guarantee the same promise, one requires a special type of promise: the promise to cooperate with neighbouring agent-nodes, about basic promise themes.*

A complete structure looks like this:

- A_1 promises p to A_3.
- A_2 promises A_1 to collaborate about p (denote this as a promise $C(p)$).
- A_1 promises A_2 to collaborate about p (denote this as a promise $C(p)$).
- A_2 promises p to A_3

By measuring p from both A_1 and A_2, A_3 acts as a judge of their compliance with the mutual agreements between them (see fig. 5.1). This allows the basis of a theory of measurement, by third party monitors, in collaborative networks. It also shows how to properly define structures in the file-directory example (see fig 5.1).

5.3.5 Example: dependencies and handshakes

Even networks of autonomous agents have to collaborate and delegate tasks, depending on one another to fulfill promised services. An important matter is to either find a way of expressing dependency relationships without violating the primary assumption of autonomy, or prove that it cannot be done[14].

Consider three agents A_1, A_2, A_3, a database server, a web server and a client. We imagine that the client obtains a web service from the web server, which, in turn, gets its data from a database. Define propositions and validities:

- $p_1 =$ "will send database data in less than 5ms", $v(p_1, A_1 \to A_2) = T$.
- $p_2 =$ "will send web data in less than 10 ms", $v(p_2, A_2 \to A_3) = T$.

These two promises might, at first, appear to define a collaboration between the two servers to provide a promise of service to the client, but they do not.

The promise to serve data from $A_1 \to A_2$ is in no way connected to the promise to deliver data from $A_2 \to A_3$:

- A_2 has no obligation to use the data promised by A_1.
- A_2 promises its web service regardless of what A_1 promises.
- Neither A_1 nor A_3 can force A_2 to act as a conduit for database and client.

We have already established that it would not help to extend the validity function to try to group the three nodes into a Kripke 'world'. Rather, what is needed is a structure that complete the backwards promises to *utilize* promised services – promises that completes a *handshake* between the autonomous agents. We require:

- A promise to uphold p_1 from $A_1 \to A_2$.
- An acceptance promise, to use the promised data from $A_2 \to A_1$.
- A conditional promise from $A_2 \to A_3$ to uphold p_2 iff p_1 is both present and accepted.

5.3.6 Acceptance

We have deduced that three components are required to make a dependent promise. This requirement cannot be derived logically; rather, we must specify it as part of the semantics of autonomy.

Axiom 2 (Acceptance or utilization promise rule) *Autonomy requires an agent to explicitly accept a promise that has been made, when it will be used to derive a dependent promise.*

We thus identify a second special type of promise: the "usage" or "acceptance" promise that we discuss in the next chapter. This is a promise to receive so it falls into the class of promise bodies denoted $-b$.

What use is this construction? First, it advances the manifesto of atomicity, making making all policy decisions explicit. The construction has two implications:

1. The component atoms (promises) are all visible, so the inconsistencies of a larger policy can be determined by the presence or absence of a specific link in the labelled graph of promises, according to the rules.

2. One can provide basic recipes (handshakes etc.) for building concensus and agent "societies", without hiding assumptions. This is important in pervasive computing, where agents truly are politically autonomous and every promise must be explicit.

The one issue that we have not discussed is the question of how cooperative agreements are arrived at. This is a question that has been discussed in the context of cooperative game theory[FB04, Axe97], and will be elaborated on in a future paper[BFb]. Once again, it has to do with the human aspect of collaboration. The reader can excerise imagination in introducing fictitious, intermediate agents to deal with issues such as shared memory and resources.

5.3.7 Temporal behaviour

As an addendum to this discussion, consider *temporal logic*: this is a branch of modal logic, in which an agent evolves from one Kripke world into another, according to a causal sequence, which normally represents time. In temporal logic, each new time-step is a new Kripke world, and the truth or falsity of propositions can span sequences of worlds, forming graph-like structures. Although time is not important in *declaring* policy, it is worth asking whether a logic based on a graph of worlds could be used to discuss the collaborative aspects of policy. Indeed, some authors have proposed using temporal logic and derivative formalisms to discuss the behaviour of policy, and modelling the evolution systems in interesting ways[BLMR04, BLMR03, LM05].

The basic objection to thinking in these terms is, once again, autonomy. In temporal logic, one must basically know the way in which the propositions will evolve with time, i.e. across the entire ordered graph. That presupposes that such a structure can be written down by an authority for the every world; it supposes the existence of a global evolution operator, or master plan for the agents in a network. No such structure exists, *a priori*. It remains an open question whether causality is relevant to policy specification.

5.3.8 Interlopers: transference of responsibility

One of the difficult problems of policy consistency is in transferring responsibilities from one agent to another: when an agent acts as as a conduit or interloper

for another. Consider agents a, b and c, and suppose that b has a resource B which it can promise to others. How might b express to a: "You may have access to B, but do not pass it on to c"?

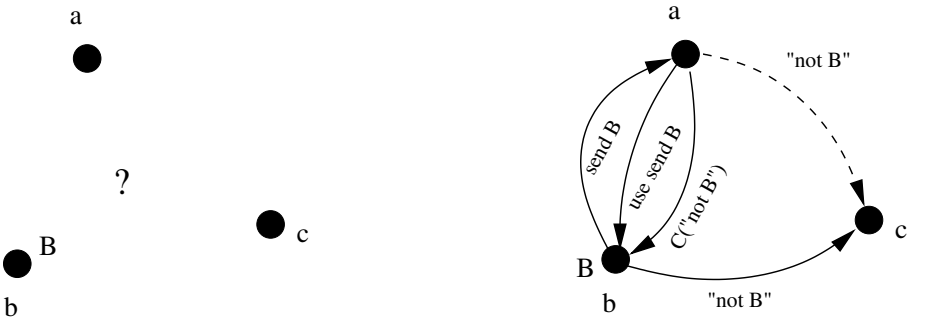


Fig. 5.2. Transference of responsibility.

The difficulty in this promise is that the promise itself refers to a third party, and this mixes link-worlds with constraints. As a single promise, this desire is not implementable in the proposed scheme:

- It refers to B, which a has no access to, or prior knowledge of.
- If refers to a potential promise from a to c, which is unspecified.
- It preempts a promise from a to b to never give B along $a \to c$.

There is a straightforward resolution that maintains the autonomy of the nodes, the principle of separation between nodes and constraints, and which makes the roles of the three parties explicit. We note that node b cannot order node a to do anything. Rather, the agents must set up an agreement about their wishes. This also reveals that fact that the original promise is vague and inconsistent, in the first instance, since b never promises that it will not give B to c itself. The solution requires a cooperative agreement (see fig. 5.2).

- First we must give a access to B by setting up the handshake promises: i) from $b \to a$, "send B", ii) from $a \to b$, accept/use "send B".
- Then b must make a consistent promise not to send B from $b \to c$, by promising "not B" along this link.
- Finally, a promises b to cooperate with b's promises about "not B", by promising to cooperate with "not B" along $a \to b$. This implies the dotted line in the figure that it will obey an equivalent promise "not B" from $a \to c$, which could also be made explicit.

At first glance, this might seem like a lot of work for express a simple sentence. The benefit of the construction, however, it that is preserves the basic principles of make every promise explicit, and separating agents-nodes from their intentions. This will be crucial to avoiding the contradictions and ambiguities of other schemes.

5.4 Modal logic and reinterpretation using promises

The usual formulation of modal logical in terms of necessity is unnecessarily fuzzy, even to the point of being incorrect. The language of promises helps to clarify what is going on here.

The problem lies in the semantics of the most basic assumptions about necessity. The rule that $\Box\, p \rightarrow \Box\,\Box\, p$ is pure nonsense if $\Box$ means necessity. It is not at all necessary that things be necessary. One can choose 'requirements' which is a voluntary act about things defined to be necessary. Conversely, one can mandate a voluntary choice in a multiple choice exam. Thus the common language interpretation is simply wrong. However, all is not lost.

Consider a reinterpretation of these quantities as follows:

$\Box\, p$	$Vp = \neg\Box\, p$
p is involuntary	p is voluntary

Involuntary acts are made by an irresistable force, so we we are led to the need to speak of forces that are beyond an agent's control. However, every one of the following possibilities might be true in some circumstances:

- $\Box\,\Box\, p$: it was forced upon the agent to make p involuntarily true (coercion).
- $V\Box\, p$: the agent chose to make p involuntarily true (discipline).
- $\Box\, Vp$: the agent was forced to make p a voluntarily choice (authority).
- VVp: the agent chose to make p a voluntarily choice (decision).

To handle this case, one can imagine making a proposition p partially voluntary, so that it consists of a voluntary (promised) part p_v and an involuntary part $p_\Box$:

$$p = p_v \cup p_\Box - p_v p_\Box \tag{5.7}$$

$$\Box\, p = p_\Box \tag{5.8}$$

$$Vp = p_v \tag{5.9}$$

Then we can say

$$p_v \cap p_{\Box} = \emptyset \tag{5.10}$$

$$Vp \cap \Box p = \emptyset. \tag{5.11}$$

5.5 The use promise is not primitive

The use-promise we have referred to so far cannot be primitive promise type, since it includes implicit information about the promise. We can express this by defining:

$$U(b) \equiv -\psi(b) \ , \ \Upsilon(b). \tag{5.12}$$

i.e.

$$\text{Use} \equiv \text{knowledge of content} \ , \ \text{intention to employ content}$$

where $\Upsilon(b)$ is the primitive promise to act on an unconditional promise $\pi(b)$ that it has (necessarily) received directly.

5.6 Transactions and duality

Each promise graph, classified in terms of $+$ and $-$ promise types has at least two dual or complementary viewpoints.

5.6.1 Complementarity: Intent vs Implementation

The first concerns the duality between planning and implementation, or declarative and imperative forms of a plan. We can see this as in fig. 5.3.

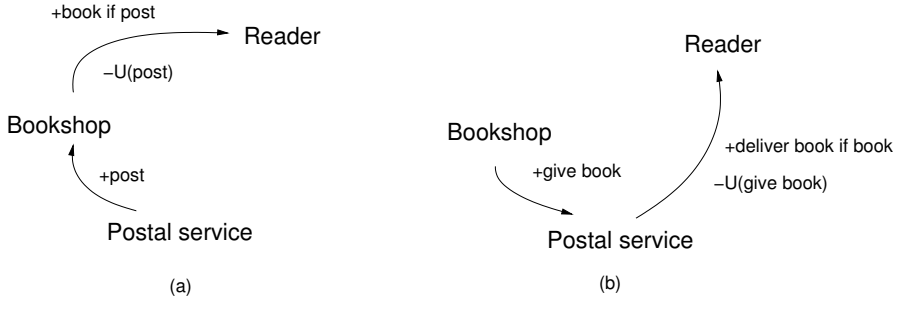


Fig. 5.3. Alternative interpretations of a service interaction, in terms of a service or transport.

In the service view (a), the service provider takes ultimate responsibilty by making a promise directly to the end reader, but it is a promise conditional on the behaviour of the post office whose role is to deliver the book. The positive aspect of this view is that it reflects the reality of the trading interaction. The post office is merely an assistant (see section 6.3.1). This is a version of causation in which the original intention is the driver for events.

In the transport view (b), we model this more closely related to the physical implementation of the promise. The service provider (bookshop) promises to pass the book to the post office, who in turn promises the reader to deliver it, assuming that it gets the book in the first place. This is a version of causation in which transactions leading to fulfillment of the promise are in focus.

In our view, the first of these is a more accurate representation of the scenario, as it provides a deeper explanation for the events that happen to transpire, and it places the end points of the service delivery in a direct relationship with one another.

5.6.2 Causation and time-reversal symmetry

Consider fig. 5.4 in its two incarnations. The left and right hand versions tell exactly the same story, with structually identical graphs, yet the $+$ and $-$ signs have been reversed. The re-interpretation suggests (but does not prove) that the reversal that takes place is the agent initiating the relationship. It is often natural to assume that a $+$ promise must come before a use-promise to make use of it. However, this shows that no such rule is necessarily the case, as by renaming the promises, we achieve the opposite result[15].

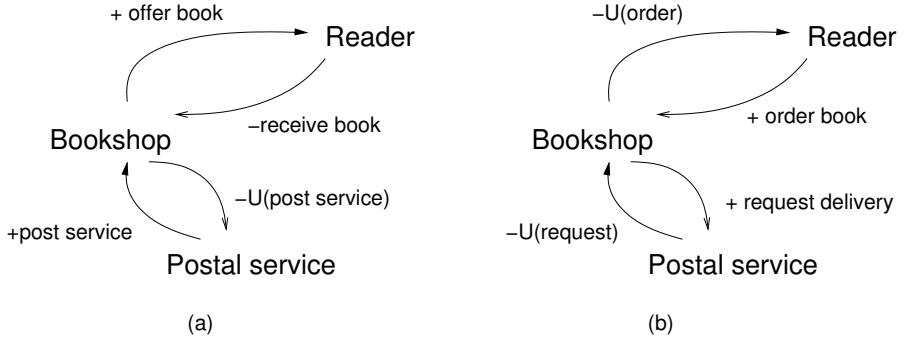


Fig. 5.4. Alternative interpretations of a service interaction, in who initiates the transaction.

This insight is part of a general symmetry in transacations between giver and receiver.

The symmetry between $\pm$ promises is a fundamental one, what physicists would call *time reversal symmetry*, or the lack of an 'arrow of time', in this case 'who goes first'. Such is it with all physical laws and mathematical expressions of change that there is no implied direction to these causative arrows initially. It is up to the analyst to break the symmetry by specifying a *boundary condition* (or, in this case, initial condition), which manifestly breaks the symmetry by deciding a given point at which we have certain knowledge of the system concerned and in which direction from this milestone the predictions (promises) of behaviour take us from that point.

We should not assume anything about which agent makes or keeps a promise first, simply from the structure of the graph. Additional information is needed to specify the direction of causation.

Notes

[14] In nearly all cases agents are working without irresistable forces guiding them, so the prospect of not being able to express cooperative tasks as voluntary autonomous cooperation has to be regarded as contrived to begin with. We claim it is 'intuitively obvious' that anything that can be achieve by force can also be achieved willingly.

[15] Indeed, we could interpret an initial promise to use a service as signalling a request for a service (like placing an advertisement).

6
Promise Mechanics

Having considered the elementary properties of μ-promises, the next step is to consider how patterns of such promises can be studied for practical application. We refer to this as 'promise mechanics', by straightforward extension of the physics of bodies. The mechanics of a system of agents may be divided into three parts:

- *Statics*: the study of equilibrium configurations of agents.
- *Kinematics*: analysis of change and other phenomena in agents.
- *Dynamics*: the study of change in relation to external influences.

In this chapter we begin with statics, and we proceed with the latter in subsequent chapters.

6.1 The laws of promise composition

6.1.1 Multiple promises by a single agent

Promise bodies are defined mathematically in terms of sets. This allows for a very general formulation of micropromises, and rather simple laws of composition by direct sum. Put simply, the promise bodies of promises between a single pair of agents combine by adding directly together. Moreover, any repetition in the bodies from multiple promises is only counted once, in virtue of rule 1.

Law 1 (Composition of promises between identical agents) *Let b_1 and b_2 be the bodies of two promises between two ensembles of agents $\{A\}_1$ and $\{A\}_2$. Then, in virtue of the assumed body semantics, and idempotence of micro-promises:*

$$\frac{\dfrac{A_1 \xrightarrow{b_1} A_2 \ , \ A_1 \xrightarrow{b_2} A_2}{A_1 \xrightarrow{b_1 \ , \ b_2} A_2}}{A_1 \xrightarrow{b_1 \cup b_2} A_2} \tag{6.1}$$

The proof of this followed trivially from the set nature of the bodies. To make this explicit, we write the promise body $\pi = \langle \tau, \chi \rangle$ and consider how these parameters combine. We can now form trivial bundles of promises between a single pair of agents by union.

$$\frac{A_1 \xrightarrow{b_1} A_2 \ , \ A_1 \xrightarrow{b_2} A_2 \ , \ \ldots A_1 \xrightarrow{b_N} A_2}{A_1 \xrightarrow{b_1 \cup b_2 \cup \ldots b_N} A_2} \tag{6.2}$$

and

$$b_1 \cup b_2 = \langle \tau_1 \cup \tau_2, \chi_1 \cup \chi_2 \rangle \tag{6.3}$$

The types τ may overlap here.

6.1.2 Ensemble promises

We can compose *bundles* of promises by union of the promisers into an ensemble.

$$\begin{aligned} A_1 &\xrightarrow{\pi} \{n\} \\ \{n\} &\xrightarrow{\pi} A_1 \end{aligned} \tag{6.4}$$

This means a promise from one agent to every member of a ensemble and vice versa.

$$\{n\}_1 \xrightarrow{\pi} \{n\}_2 \tag{6.5}$$

This means a promise from every agent in one ensemble to every agent in another. We can write

$$\odot(\{n\}_1) \xrightarrow{\pi} \odot(\{A_2\}) \tag{6.6}$$

to mean one agent from the ensemble $\{A_1\}$ makes a promise to one agent from the ensemble $\{A_2\}$.

6.2 Bindings

Suppose one agent promises τ_1, χ_1 but the recipient of the promise accepts only τ_2, χ_2. The transferred effect of the promise binding is then given by the overlap or intersection of what is offered and what is accepted:

$$\frac{a \xrightarrow{\langle \tau_1, \chi_1 \rangle} b, a \xrightarrow{U(\langle \tau_2, \chi_2 \rangle)} b}{a \xrightarrow{(\langle \tau_1 \cap \tau_2, \chi_1 \cap \chi_2 \rangle = \langle \tau_1, \chi_1 \rangle)} b} \tag{6.7}$$

Or, put another way

$$\frac{a \xrightarrow{\langle \tau_1, \chi_1 \rangle} b, a \xrightarrow{U(\langle \tau_2, \chi_2 \rangle)} b}{a \xrightarrow{(\langle \tau_1 \subseteq \tau_2, \chi_1 \subseteq \chi_2 \rangle} b} \tag{6.8}$$

The implied semantics here are to say that a conditional promise is kept if *at least* the condition is kept. i.e. we don't need an exact match, since the condition is an enabler, not a constraint.

A binding or handshake may be described by the asymmetric notation $A_1 \overset{b}{\leftrightarrow\!\!\!\!\rightarrow} A_2$, which is an irreversible replacement rule defined as follows.

$$\frac{A_1 \xrightarrow{\langle \tau_1, \chi_1 \rangle} A_2, A_2 \xrightarrow{U(\langle \tau_2, \chi_2 \rangle)} A_1}{A_1 \overset{\langle \tau_1 \subseteq \tau_2, \chi_1 \subseteq \chi_2 \rangle}{\leftrightarrow\!\!\!\!\rightarrow} A_2} \tag{6.9}$$

The arrow implies that the promise of something is being offered from A_1 to A_2 and there is acceptance in the opposite direction[16].

What if the use-promise does not match the conditional promise? Then the promise is still conditional unless the condition is met, i.e. the intersection of the guarantee and the condition includes the full condition.

6.3 Assessable promises with conditions attached

Not all promises are made without attached conditions. For instance, we might promise to pay for something 'cash on delivery', i.e. only after a promised item has been received. Such promises will be of central importance in discussing processes and trading later in this book. The truth or falsity of a condition C may be promised by an agent that is able to assess the condition as follows.

$$A_1 \xrightarrow{\mathrm{T}(C)} A_2 \tag{6.10}$$

is a promise from A_1 to A_2 of accurate information that the condition C holds. Similarly,

$$A_1 \xrightarrow{\mathrm{F}(C)} A_2 \tag{6.11}$$

is a promise from A_1 to A_2 of accurate information that the condition C is false.

A promise with body b made conditionally on such a Boolean condition C is now written in a notation reminiscent of conditional probabilities, as:

$$A_1 \xrightarrow{b|C} A_2. \tag{6.12}$$

Moreover, since the condition is merely a promise itself, we can generalize this notion of conditionals to include any kind of promise that must be kept to satisfy the pre-requisite.

Definition 42 (Conditional promise) *Let b be the promise body, and X be a pre-requisite promise must be kept before b can be kept, then we denote the promise of b given X also by*

$$A_1 \xrightarrow{b|X} A_2. \tag{6.13}$$

6.3.1 The Laws of Conditional Assistance

To make conditional promises work in an intuitive way, we need some basic assumptions. We assert that a promise that is made conditionally is not an assessable promise, unless the condition that predicates has also been promised. Further, a conditional promise is exactly 'empty' if it is predicated on something that is known to be false. Any other case simply cannot be assessed and is hence null-potent. These are rules of logic that complete the rules of set composition.

Rule 6 (Exact quenching of conditionals) *In a promise made conditionally on C, promising simultaneously the the truth of condition leads to an unconditional promise of the full magnitude:*

$$\frac{A_1 \xrightarrow{b|C} A_2,\ A_1 \xrightarrow{\mathrm{T}(C)} A_2}{A_1 \xrightarrow{b} A_2} \tag{6.14}$$

Conversely, when the condition is false, the remaining promise is rendered empty:

$$\frac{A_1 \xrightarrow{b|C} A_2,\ A_1 \xrightarrow{\mathrm{F}(C)} A_2}{A_1 \xrightarrow{\emptyset} A_2} \tag{6.15}$$

Note that, since promising $T(C)$ and promising b are two different types of promise, this rule does not follow directly by the normal set algebra in section 6.1. We must deduce it from the fact that both of these promises are: (i) made to and from the same set of agents, and therefore (ii) calibrated to the same standards of trustworthiness. Thus, logically we must define it to be so.

We can generalize these results for promises of a more general nature, by the following rules.

Rule 7 (Exact quenching of pre-requisite) *If a promise with body S is provided subject to the provision of a pre-requisite promise X, then the provision of the pre-requisite by the same agent is logically equivalent to the unconditional promise being made:*

$$\frac{A_1 \xrightarrow{S|X} A_2 \ , \ A_1 \xrightarrow{+X} A_2}{A_1 \xrightarrow{S} A_2} \tag{6.16}$$

Indeed, we define this to be an identity:

$$A_1 \xrightarrow{S} A_2 \equiv A_1 \xrightarrow{S|X} A_2 \ , \ A_1 \xrightarrow{+X} A_2 \tag{6.17}$$

This gives us a rewriting rule for promises made by a single agent in the promise graph. The $+$ is used to emphasize that X is being offered, and to contrast this with the next case.

Consider now the first case in which one agent assists another in keeping a promise. According to our axioms, an agent can only promise its own behaviour, not that of other agents, thus the promise basically comes from the principal agent, not the assistant, which might not even be known to the final promisee[17]. Assistance is purley a matter of voluntary cooperation on the part of the tertiary agent.

Once again, we cannot derive the following, we merely define it to be so:

Rule 8 (Assisted quenching of pre-requisites) *If a promise with body S is provided subject to the provision of a pre-requisite promise X, then the provision of the pre-requisite by an assistant is acceptable if and only iff the principal promiser also promises to acquire the service X from an assistant (promise labelled $-X$):*

$$A_1 \xrightarrow{S} A_2 \sim A_3 \xrightarrow{+X} A_1, A_1 \xrightarrow{S|X} A_2 \ , \ A_1 \xrightarrow{-U(X)} A_2 \tag{6.18}$$

We refer to this as an *assisted promise*. The principal A_1 promises both that it will keep S provided X is given, and that it will both procure and use X. Then to complete the picture, a third party A_3 must provide X to A_1.

This gives us another rewriting rule, though not a true equivalence this time, because there is the introduction of a third party, whose skills and properties are not known by the agent A. However, we can define it to be thus, understanding that the responsibility belongs with agent A_1 if A_3 does not deliver.

Henceforth, for brevity, we define the notation:

$$A_1 \xrightarrow{S|X} A_2 \ , \ A_1 \xrightarrow{-U(X)} A_2 \equiv A_1 \xrightarrow{S(X)} A_2. \tag{6.19}$$

6.3.2 Chaining promises - intermediaries and logistics

Promise chains are sequences of agents that assist one another in the keeping of a promise. Promise chains are important in workflow processes and end-to-end delivery problems such as data transmission. As we can already see from the foregoing sections, assisted promises are non-trivial. There is a whole spectrum of achievable levels of certainty depending on how many promises we can obtain from the different parts of the chain. We return to this issue in section 12.4.

6.3.3 Transitivity of promises

Promises are not transitive in general, but agreement is transitive.

Describe the approach to

6.3.4 Transmission and distortion of information

As an addendum, we note that whenever an intermediate agent handles information and passes it on, in the manner of a relay, the integrity of the information cannot be guaranteed. This follows from the autonomy of the agents. Basically, no other agent is necessarily privy to what other promises this agent might have made. Only a promise of direct communication with the course can provide certainty in data transmission. Refer to sections 3.6.5 and 3.6 for more on this.

6.4 Coordinated behaviour

We have established that when autonomous agents make promises, they do so without any *a priori* regard for one another. Their promises do not have any notion of global consistency, nor form a complementary whole. This is both the advantage and the price of autonomy. For management, we are interested in understanding what promises are needed to make nodes behave according to a standard. To understand the incentives for promising something standard we must look at the economics of the network[BFb]. For the time being, however, we shall merely assume that coordination is a desirable property between such agents and we seek a mechanism for allowing agent nodes to agree without violating the assumption of autonomy.

The meaning of coordination is to be observably similar to a third party. Consider the three nodes in fig. 6.1. On the surface, a promise made by node A_2 to node A_3 does not imply any constraint or information about a promise from node A_1 to node A_3: the nodes are autonomous. But, what if we introduce

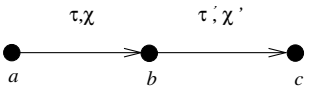


Fig. 6.1. Serial composition of promise and subordination, leads to limited transitivity. The dashed arrow is implied by the $C(b)$ promise.

an additional promise from A_1 to A_2? Then, we can ask node A_1 to coordinate with node A_2 about its behaviour towards A_3. This requires additional promises. We shall often use a shorthand notation for a *coordination* (or subordination) promise, denoting its body $C(b)$. This is the promise to do the same as another agent with respect to a promise body b. This is not a primitive type, but we denote it by the shorthand:

$$A_1 \xrightarrow{C(b)} A_2 \tag{6.20}$$

This means that A_2 informs A_1 about its promises of type b, and the receipt and usage of that information by A_1. This promise is a subordination for the agent because A_1 is willingly giving up its autonomy in the matter of b by agreeing to follow A_2's lead. We might call this a 'slave' promise.

Definition 43 (Coordination/subordination promise) *Let $\psi(b)$ mean 'knowledge about the promises with body b. We define the coordination promise:*

$$A_1 \xrightarrow{C(b)} A_2 \equiv \left\{ \begin{array}{c} A_2 \xrightarrow{+\psi(b)} A_1 \\ A_1 \xrightarrow{-\psi(b)} A_2 \\ A_1 \xrightarrow{\Upsilon(b)} A_2 \end{array} \right. \tag{6.21}$$

The slave agent promises to do the same as the master agent. This requires the master node to inform the slave of its intentions and for the slave to agree to comply with these.

Notice that, by agreeing to follow another agent's promises, this is also a voluntary subordination of the agent. It is not an obligation, because the promise can easily be withdrawn, but it is as close as we can get voluntarily.

We may now write an equivalent promise for the case in which A_1 promises A_2 that it will coordinate on the matter of b, given that A_2 promises A_3 b, and

we use the symbol " , ", as above, to signify the composition of these promises.

$$\underbrace{\overset{C(b)}{A_1 \to A_2}}_{\text{'Coordinate with'}} \quad , \quad \underbrace{\overset{b}{A_2 \to A_3}}_{\text{Promise}} \quad \Rightarrow \quad \overset{b}{A_1 \to A_3} \; . \tag{6.22}$$

This allows us to deduce something about a 'global consistency' of interaction with a third-party, such as might be observed by an external observer of the agents with global knowledge. We see that this combination of promises implies that there ought logically to be a promise of type b from A_1 to A_3. It has the appearence of transitivity, but it is more subtle. In this example, we have one agent following the other. If the first agent changes its mind, then both promises can drift together like a 'swarm'[BF07]. To remove this privileged leadership, we can symmetrize the promises (fig 6.2):

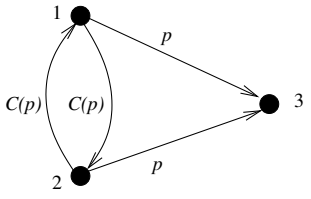


Fig. 6.2. Coordination and the use of third parties to measure the equivalence of agent-nodes in a region. Agents form groups and roles by agreeing to coordinate policy.

If two nodes pledge their promise to an external party (a form of allegiance), then they have made an implicit agreement with one another also, which binds them together. This result is important for seeing the actual groupings of roles and structures in a policy graph (see proof in appendix).

A complete structure, using the shorthand cooperative notation, looks like this:

- A_1 promises p to A_3.
- A_2 promises A_1 to collaborate about p (denote this as a promise $C(p)$).
- A_1 promises A_2 to collaborate about p (denote this as a promise $C(p)$).
- A_2 promises p to A_3

The cooperation promise will turn out to be an importance point of contact with the theory of games and voluntary cooperation[Axe97, Axe84]; this is discussed in a separate paper[BFb].

6.5 Subordination and autonomy

Subordination is a partial ordering on agents. Define the ordering operator

$$A <_{\text{sub}} B \tag{6.23}$$

meaning that agent A is subordinate to agent B. Then,

$$A <_{\text{sub}} B \;\rightarrow\; A \neq B \tag{6.24}$$
$$A \leq_{\text{sub}} B \;\leftrightarrow\; (A = B) \vee (A <_{\text{sub}} B) \tag{6.25}$$
$$A \not<_{\text{sub}} A \tag{6.26}$$

And agent A is autonomous if it is the maximal element in this ranking, i.e. an arbitrary agent a satisfies $\forall a : a \leq_{\text{sub}} A$.

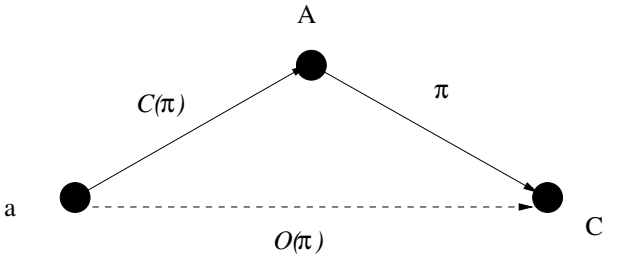


Fig. 6.3. Voluntarily simulated obligation.

An agent cannot voluntarily give up its autonomy, since it is free to revoke such a promise, but we can easily simulate subordination voluntarily.

Definition 44 (Subordination promise) *The so-called coordination promise introduces a voluntary obligation by giving up autonomy in the matter of a promise body $\pi = \langle \tau, \chi \rangle$. We already have a way of making something like an obligation in the promise theory. This is a limited undestanding of an obligation within the promise framework. Let A be an authority and a, C be agents.*

$$\frac{a \xrightarrow{C(\pi)} A \;,\; A \xrightarrow{\pi} C}{a \xrightarrow{O(\pi)} C} \tag{6.27}$$

Then by subordinating to A's wishes, a becomes obliged to promise π to C. However, $a \not<_{\text{sub}} C$.

6.5.1 Obligations

Obligations are 'deontic' modal statements like: X must comply with Y, A should do B, C is allowed to do D, and so on. They assume that one can force a system to bow down to a decision made externally. Obligations can lead to pitfalls and complications in modelling as they lead to distributed inconsistency (see chapter 2).

Obligations are imposed on an agent from outside, i.e. by exteral agents. Autonomous agents are, by definition, never obliged to do anything they have not decided for themselves. To accept an obligation, an autonomous agent must sacrifice some of its autonomy. This assumes the existence of an *authority*.

Assumption 2 (Obligation requires three parties) *There are three agents involved in an obligation: an agent imposing the obligation (an authority), and agent that is obliged to promise something to another and an agent and the receiving agent.*

For example, the law obliges me to promise not to steal from you. We would like to avoid the notion of obligation, since it is problematic. Obligations to an agent could be made from many distributed locations, so consistency is unlikely. Each agent's promises are all made in one place, so inconsistencies are more easily uncovered.

We can use the same bodies to talk about obligations.

$$a \xrightarrow{O:b} c \tag{6.28}$$

means that a is obliged to comply with b for c. How does this differ from a promise?

- a might not know about an obligation, but it is always aware of its promises.
- c might not know about the obligation.
- An obligation comes from a different agent than the one obliged.
- An obligation cannot be withdrawn by a, nor was it declared by a.

We might want to compare something like the following rule, as a simulation of an obligation (see fig 6.3).

$$\frac{a \xrightarrow{\pi \notin K} b}{a \xrightarrow{O(\pi)} b} \tag{6.29}$$

where $O(\pi)$ is an implicit self-imposed obligation derived from the voluntary promise. i.e. any promise that is not automatically kept places a voluntary 'obligation' on the promiser. This is not the most general idea of an obligation, because it doesn't show how autonomy is sacrificed. It applies in the sense that the obligation was imposed by oneself, and since this is both redundant and does not enhance our discussion of promises we shall not discuss this further.

6.6 Orchestration, subordination and coordination

Subordination is where one agent agrees to follow the lead (orders) of another (see section 6.4). This is not a fundamental promise type – we can make it from give/take promises. However, it is convenient shorthand.

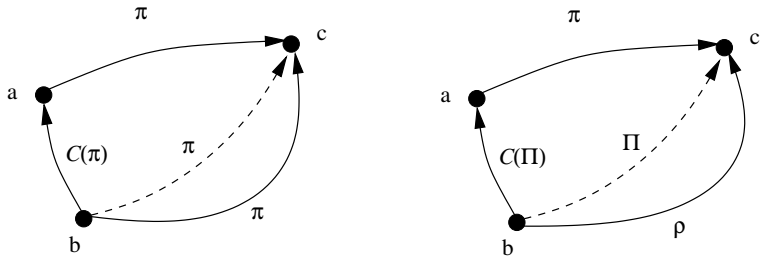


Fig. 6.4. A coordination promise (implying a subordination) implies a promise from one agent to another (dotted line). If coordination promises do not agree with actual promises, then we see this as a broken promise from the agent b.

See fig. 6.4. When an agent promises to follow another agent's example, it can optionally limit its compliance. Suppose an agent sets an example by promising:

$$a \xrightarrow{\pi=\langle\tau_1,\chi_1\rangle} c \tag{6.30}$$

A second agent b can promise to follow a using a coordination promise $C(\langle\tau,\chi\rangle)$:

$$b \xrightarrow{\langle\tau_1,\chi_1\rangle} a \tag{6.31}$$

which would have the immediate consequence:

$$\frac{a \xrightarrow{\langle\tau_1,\chi_1\rangle} c, b \xrightarrow{C(\langle\tau_1,\chi_1\rangle)} a}{b \xrightarrow{\langle\tau_1,\chi_1\rangle} c} \tag{6.32}$$

This is one of the few examples of transitivity in promise theory. But what if agent b does not entirely agree to follow agent a? He wishes to retain some control, or set his own limits. Then the resulting implied promise is an intersection:

$$\frac{a \xrightarrow{\langle \tau_1, \chi_1 \rangle} c, b \xrightarrow{C(\langle \tau_2, \chi_2 \rangle)} a}{b \xrightarrow{\langle \tau_1 \cap \tau_2, \chi_1 \cap \chi_2 \rangle} c} \tag{6.33}$$

If the agent should further wish to deviate from this by promising an amount that deviates from this intersection, then the implied promise would combine with that promise to form a broken promise:

$$\frac{b \xrightarrow{\pi_1 \cap \pi_2} c, b \xrightarrow{\pi_3} c}{\text{Broken promise} : \pi_1 \cap \pi_2 \neq \pi_3} \tag{6.34}$$

Or in fig. 6.4 notation:

$$\frac{b \xrightarrow{\pi \cap \Pi} c, b \xrightarrow{\rho} c}{\text{Broken promise} : \pi \cap \Pi \neq \rho} \tag{6.35}$$

6.7 Cooperation and dependence between agents

Consider a simple model of interaction that we shall return to throughout the book (see also section 12.4). Cooperation implies that agents coordinate their promises for mutual gain. Cooperation is thus more than coordination: it has an economic basis.

We may define cooperation to be the keeping of promises that support or align with the goal of keeping this economic relationship in place. What does it mean for agents to align with a goal to provide S? We may assume that the agent B has the goal of providing the service S, and as the originator of this goal, thus it is aligned with it by definition, but what about other cooperating agents? To more general cooperation, we may introduce an intermediate agent I in the execution of the service.

We can display this as the workflow diagram shown by the dotted lines in fig 6.5. The business B would like to provide service S to its customer C; in actuality this requires the help of intermediary I. The cooperation of this agent I must lead to value if it is aligned with the goal of delivering S. Goals can be considered promises that that yield positive value to an agent in any abstract currency.

It is unimportant to whom goal promises are made, or even what the currency of remuneration is. For a business, some of them are promises to business associations, some are perhaps promises to self or to society at large. Others might

be promises to the law courts, or whatever representative agent is the considered custodian of litigious correctness. All that is important from the viewpoint of promise theory is that the 'business agent' is linked to some other agent by such a promise, and that the business agent itself places a value on that promise, whether it be coming or going, positive or negative.

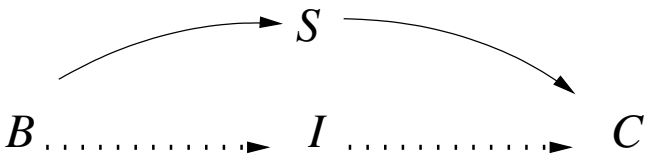


Fig. 6.5. Inserting an intermediate agent into a business process. The dotted lines show a work flow path. The arc shows a promise the business would like to make to the end customer – but promise theory says that it cannot if it does not have direct contact.

The flow of work must now proceed through a chain of intermediary promises that includes the agent I. From the perspective of promises, an intermediate agent's alignment will be judged by the agent B if it keeps a promise that confers value to B's service promise.

- A promise might be a "make or break" i.e. an *enabler* without which the service S cannot be provided.

$$I \xrightarrow{\rho} B \xrightarrow{S/\rho} C \tag{6.36}$$

  By promising to provide ρ, I aligns with B's promised goal to provide service S.

- A promise is more often an *effector* or amplifier which parametrically increases business result.

$$I \xrightarrow{\rho} B \xrightarrow{S(\rho)/\rho} C \tag{6.37}$$

  By promising to provide ρ, I aligns with B's promised goal to improve service S.

- Any miscellaneous promise is aligned with business goal if it confers a positive value to the business agent B according to its own judgment.

$$v_B\left(\text{Any} \xrightarrow{b} B\right) > 0. \tag{6.38}$$

We may therefore conclude that cooperation is the keeping of a promise that is viewed as valuable to another agent. A goal is a promise that brings value, and alignment with a goal implies keeping a promise that supports the successful keeping of the goal promise.

6.8 Deadlock in conditional promises

Definition 45 (Conditional promise to act) *A statement by an agent A to another agent A' that it has decided to perform an action a at some time after it learns the truth of a condition c.*

Suppose now that two agents are involved in a statement of willingness to perform two independent actions A_1 and A_2 that are linked through a conditional, namely that one is predicated by the other. e.g. "I am willing to perform A_1 if I get A_2". This statement of willingness could also expressed "If I get A_2, I am willing to perform A_1". There is no causal ordering in these actions. The temporal ordering is symmetrical as far as the preference of the agent is concerned.

To understand the scenario better, we can imagine that the agents are bargaining and A' says "I am willing to perform A_2 if I get A_1". Even if the agents' turn these expressions of willingness into promises, there is a conundrum. The two agents express complete symmetry and there can be no causal outcome. In fact the situation is a deadlock.

In order to break this deadlock, i.e. for something to happen, or for a meaningful promise to be made, we have to break this symmetry. One of the the agents must make an unconditional statement (promise) or action, i.e. it must recind its condition and simply act. This then allows the trade of promises to be made. We shall use this in chapter 7 to explain time and deviation from equilibrium.

The occurrence of unconditional promises to act suggests that all promises to act would begin with the expression of a conditional promise, which then motivates an unconditional promise in return from the counter-agent to release the preferred course of action.

6.9 Entangled promises

Correlated promises that preserve a global symmetry.

Notes

[16] One might want to call this construction an interface[BP06].

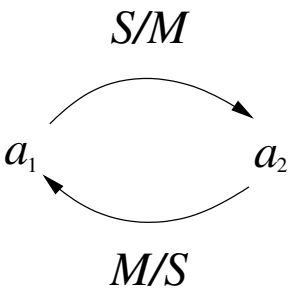


Fig. 6.6. A simple exchange of actions between agents results in a causal symmetry. There is temporal equilibrium and the preferences are balanced as long as the actions are valued by one another. This is the basis of trade. One agent must concede its condition and act unconditionally to set the initial condition that generates a causal evolution.

[17]When a conditional promise is made and quenched by an assistant, the 'contact' agent is directly responsible by default. We shall refine this view with alternative semantics later, since this is all a matter of managing the uncertainty of the promise being kept. As soon as we allow rewriting rules of this basic type, it is possible to support multiple solutions for bringing certainty with graded levels of complexity.

7
Promise Dynamics

This chapter develops the model of promises further to include a description of how agents change. It will lead us to the associated concept of behaviour. Change is the most elementary consideration in any description of a world, whether it be human, economic or mechanical. It allows us to define variation, which in turn allows us to differentiate conditions and make measurements. Our description of change will be formal and instantly recognizable from natural science, but we shall use it to study the kinematics and dynamics of systems from the viewpoint of promises.

From a philosophical perspective, this chapter takes an important step in modelling. It begins from a dispassionate and mechanistic physics of phenomena, and extends that view slightly to admit human *motivation*, or *semantics*. It does so by using the promise concept and without undue violence to the formulation[18].

One satisfying aspect of this chapter is thus to present a model that applies equally to inanimate entities as it does to living ones. By replacing context specific modelling with general notions, we end up with an approach that can be applied in equal measure to the natural sciences or the social sciences.

7.1 The promise lifecycle

The promise lifecycle refers to the various states through which a promise passes, from proposal.

Promise State	Description
proposed	A promise statement has been written down but not yet made.
issued	A promise has been published to everyone in its scope.
noticed	A published promise is noticed by an external agent.
unknown	The promise has been published, but its outcome is unknown.
kept	The promise is assessed to have been kept.
not kept	The promise is assessed to have been not-kept.
broken	The promise is assessed to have been broken.
withdrawn	The promise is withdrawn by the promiser.
expired	The promise has passed some expiry date.
invalidated	The original assumptions allowing the promise to be kept have been invalidated by something beyond the promiser's control.
end	The time at which a promise ceases to exist.

Definition 46 (Promise lifetime) *The time interval spanned from the issuance of the promise to the end of the the promise.*

Rule 9 (Promise withdrawal requires notification) *The withdrawal of a promise by an agent requires the notification of other agents in scope. Note however that if an intention is withdrawn without its corresponding promise, the promise becomes a deception.*

Rule 10 (A promise ceases to exist when broken or invalidated) *Once a promise is broken or otherwise enters one of the end states (invalidated, expired, etc), its lifecycle is ended, and further intentions about the subject must be described by new promises.*

The lifecycle of a promise may now be viewed from either the perspective of the agent making the promise, or from the perspective of the promisee, or in fact any another agent that is external but in scope of the promise.

Here the $+$ sign denotes alternative choices. The $*$ notation here means that one of the options in the parenthesis may be repeated any number of times during the lifetime of the promise. In other words, a promise's status might be re-assessed from kept to not-kept and back again any number of times. Note also that a promise that is never issued ends before its life begins.

From the perspective of the promisee, or other external agent in scope, we have a similar lifecycle, except that the promise is first noticed when published by the promiser:

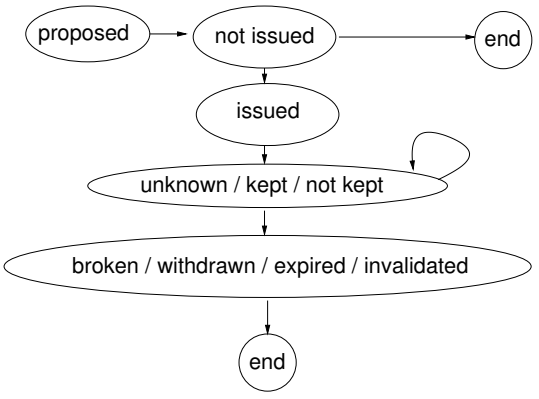


Fig. 7.1. The promise lifecycle for a promiser

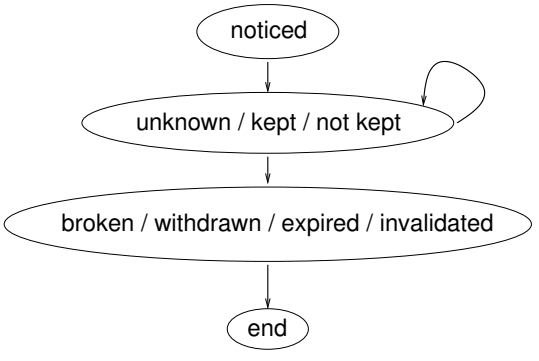


Fig. 7.2. The promise lifecycle for a promisee

7.2 Configuration space

Every systematic description of the world is a balance between allowed freedoms and imposed constraints. Systems exist and change within a given realm of possibility called a configuration space that desribes the 'degrees of freedom' of the system. This realm is usually described by a set of variables representing the independent characteristics that distinguish it, such as position in space, or a colour in the spectrum, etc. These variables can exist in a number of *states*. However, within this realm of maximal possibility, there is usually a more limited region occupied by the system.

The extent of this actual region is dictated by *constraints* upon it. Some of these constraints may be considered external (like boundaries) and some are internal (like tethers). Some are voluntary and some are involuntary. These

distinctions are not always important, and can often be unified by thinking of the world in terms of different kinds of promises.

A child that promises not to play in the road makes a voluntary constraint. Other constraints seem more fundamental. For example, in astronomy the realm of possibility is the universe, with space and time variables. For a game of squash, it is within the boxed four walls of the court. Then there are kinematical constraints, often called 'equations of motion' for the states that represent the allowed ways that a system evolves within this bounded region. For the game of squash this would be a combination of the laws of physics (which cannot be avoided) and the rules of the game (which are kept voluntarily).

As modellers we have a choice: do we represent these boundaries and limitations as inevitable facts, or do we call them implicit promises to obey laws on a voluntary basis? In many cases the distinction is irrelevant as long as we maintain the same formal book-keeping of the constraints. The distinction is mainly important to observers who attempt to assess motivations – and this applies mainly to human sciences. We can always model something involuntary as something seemingly voluntary that is stubbornly reliable or persistent in its choice[19].

Sometimes we make modelling choices about where to draw the line between what is internal or external to a problem. There is some degree of arbitrariness to these choices. Usually one makes a choice based on pragmatism; e.g., In the natural sciences, we often try to limit the appearance of the outside world by talking about idealized or closed systems, by which we mean that we intend to exclude all distractions and isolate one specific issue. This is a convenience that can bring great pedagogical simplification, allowing one to see simple principles at work, thus it is not to be frowned upon. It is important not to confuse models with an objective 'truth' however.

7.3 Change

Implicit in several the examples we have looked at so far is a description of how change occurs, from the perspective of promises and outcomes. Let us now complete this picture, with a model of states. To set the stage for the discussion, we shall begin by assuming:

- *Each agent's condition in the environment can be characterized by its state.*

  The state of an agent can include internal variables and knowledge, and it can represent outward (observable) characteristics. An agent is assumed

to be aware of its own state. In fact we want it to be in a position to make value judgements or *preferences* about the different states.

- *An agent's state changes in response to events.*.

  Changes can be either continuous or discrete in principle, but we shall focus on a discrete time theory in which there is a minimum measureable time interval. A change can be viewed as the effect of an event e. An event now needs to be defined more carefully.

- *Events will be assumed to be 'instantaneous'*. In other words, an event persists for only a single unit of the minimum time interval and then it is over.

7.3.1 Events

Definition 47 (Event) *A transition from one state to another that is observed by an agent.*

7.3.2 Encoding change using states

To distinguish change we require a way of measuring it. For this we need to introduce states. States are amongst the most primitive things in our model of change. Promises and assessments do not necessarily need to refer to states, but the outcomes of promises will lead to physical consequences in the real world, and for that we must have names for the different conditions in which a system finds itself.

Since states are familiar in many branches of science, we shall not dwell too much on the their philosophy. A state is simply a label which represents a unique configuration of the system at hand. We can regard states as primitives, since we need some essential notion of a system by which to form a measurable scale of change[20].

The set of states may be either ordered (or not) to represent the specific properties required[21].

We shall use the vector notation $\vec{q}$ for states. A state may be written in transpose forms:

$$\vec{q}_{\rm label} \ \text{ or } \ \vec{q}^{\rm T}_{\rm label}. \tag{7.1}$$

The set of labels may be quite arbitrary, but it is also useful to make use of numbers, especially when ordering is required. We henceforth represent 'ob-

servables' as vectors of variables that may take one or more values in a clearly defined set.

Since we are free to create a code-book that transforms any system of labels into numbers, simply by counting[SW49], we shall use numerical labels where convenient, as this will allow us to step through the possibilities by simple arithmetic, from an arbitrary orgin q_0 to $q_1 \ldots q_n$. We introduce step-operators that count through these states, up to a normalization factor[22] that we shall not define here.

$$a_+ \cdot q_0 \to q_1 \tag{7.2}$$

$$a_- \cdot q_1 \to q_0 \tag{7.3}$$

$$a_+ \cdot q_n \to q_{n+1} \tag{7.4}$$

$$a_- \cdot q_n \to q_{n-1} \tag{7.5}$$

The operators $a_\pm$ are sometimes called creation and annihilation operators on the numerical 'value' of the state[23]. We note that when assessing a change, an observer cannot speak of the reason or cause of a change in general, since that information lies inside the agent whose details are inaccessible.

7.3.3 Equilibrium

Promise equilibria may be viewed as pairs of bilateral promises between agents. The phenomenon is like symbiosis in biology. This mutual closure between promises is a basic topological configuration that allows for the persistence of an operational relationship. When this 'trade' of promises is stable over some time, the result is equilibrium.

Equilibrium does not imply static fixture. Dynamic or statistical equilibria describe properties that are in balance on average. This is the more normal state of affairs, since noise from the environment can never be completely shielded.

Equilbria can, themselves change, if the point of balance between what is given and received changes. Slow changes in the properties of the agents or the environment can lead to a drift in the average values. If this drift is slow enough to be distinguished from the actual fluctuation exchanges themselves then we may call it *adiabatic*. This means that there is weak enough coupling between the fluctuation process and the process leading to average drift to lead to a clean separation of time scales.

Systems that have strong coupling do not exhibit this property and are much less predictable as a result[McC03, BA99]. The interaction of scales is a vast

topic that cannot be given a fair treatment here. Suffice it to say that this is a crucial part of behavourial description in any system and the promise binding description allows us to understand this in a classic interaction viewpoint.

It is easy to show that, for agents with preferences, they correspond to Nash equilbria of two-person games[24].

Ironically, equilibria are a convenient way of summarizing the information about change in a system. Suppose we collect all the different ways in which a system can be in deadlock into a series. This series is called the *effective action* of the system[Rei65, Abb92, Bur02]. It consists of promise graphs that are stalemated in increasingly complex ways. Keeping any one of the promises first in a deadlock, i.e. by breaking the symmetry along one of the interfaces, will lead to action in the system, with a definite arrow of time.

7.3.4 From equilibrium to an arrow of time

Consider a generator of a cycle (see fig. 7.3)[25], as discussed in section 6.8. We can express this more carefully now, in the language of assessments.

$$a_1 \xrightarrow{\pi_S : S/A(\pi_M)} a_2 \tag{7.6}$$

$$a_2 \xrightarrow{\pi_M : M/A(\pi_S)} a_1 \tag{7.7}$$

In the previous formulation of our model for cooperation, agent a_1 promises a service S if it assesses a promise of money M to be kept. Agent a_2 promises a_1 money M if it assesses that it has received service S. This scenario represents a 'deadlock' of the kind well know in commerce. Neither party can proceed until one of them breaks the stalemate. The model is more general than just a

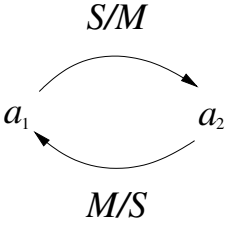


Fig. 7.3. A 'cycle' generator is two deadlocked promises back to back, waiting for the deadlock symmetry to be broken.

business interaction, of course. It could describe any kind of pump or process. Consider a car engine. There is a part that injects fuel if a flap is opened. Then

there is a part that ignites it (creating motion) if the fuel is injected. These two processes are usually in deadlock, and have two possible equilibria:

- Nothing happens – both the fuel injector and the ignition are at rest and remain so.
- Both parts are in motion in a continuous cycle.

How does the system make a transition from one equilibrium to another? Something must break the symmetry. Normally an external electric starter motor initiates motion to break this deadlock. Sometimes drivers will 'bump start' the car by pushing it and releasing the clutch if the battery is dead. In either case, there is an act of *symmetry breaking*.

Thus, to resolve the dilemma of how to get the pump started, the symmetry between the players must be broken by an 'initial condition'. Depending on which agent invokes this initial condition, one might assess one of two possible patterns of outcome:

$$\pi_S : A(\pi_S)A(\pi_M)A(\pi_S)A(\pi_M)\ldots \tag{7.8}$$

$$\pi_M : A(\pi_M)A(\pi_S)A(\pi_M)A(\pi_S)\ldots \tag{7.9}$$

In shorthand, we may write:

$$\pi_S : SMSMSMSM\ldots \tag{7.10}$$

$$\pi_M : MSMSMSMS\ldots \tag{7.11}$$

Selecting one of these sequences is like selecting the 'arrow of time' in the system – deciding in which direction the cycle proceeds, because reversing the order of who goes first looks very much like making the loop go clockwise or anticlockwise, depending on who goes first. In the long run, it does not matter one way or the other, as once the memory of the initial condition is long gone, the sequences are indistinguishable – they visit all the same states in equal number[26].

It is a question of symmetry breaking[27].

7.3.5 Broken symmetry

1. A broken symmetry that selects a direction from some arbitrary boundary value.
2. A set of states that may be used to label or measure the change.

The arrow of time appears in promises through *pre-conditions*. A conditional promise to act

$$A \xrightarrow{b|c} B \tag{7.12}$$

with body b and precondition c leads to an ordering of b and c.

7.4 Effective action of a promise

The set of such $\pm$-loop processes is called the effective action of the system of ensemble of agents. It plays the role of a generator of behaviour. To begin with, there is no broken symmetry in this generator so the behavioural outcome of the promises cannot be known until a specific initial state is specified (see section 5.6.2).

....

7.5 Voluntary and involuntary constraints

7.5.1 Behaviour versus control

In technological disciplines, it is common to assume that the outcome of a system is to behave according to its design. In natural science, this viewpoint is usually absent as the universe is assumed to be an emergent outcome of processes that were not planned. Promises allow us to capture both of these viewpoints and find a reasonable forum for them to interact.

The behaviour of a system is not only about what is planned or desired, but about what actually happens when a system develops in its environment. This includes many unpredictable factors that need to be described in a realistic model.

7.5.2 Force, attack and involuntary change

One of the issues which voluntary cooperation de-emphasises is the notion of action by force. One can reason for this point of view in a number of ways, moral, political, practical etc. However, the concept of force is necessary; the perception of force does exist in nature and must therefore be modelled.

The purpose of voluntary cooperation is not to reject such ideas of physics, but to abstract its processes in form that allow one to see similarities and principles more clearly, while rejecting distinctions that are merely arbitrary. A theory of agents must therefore acknowledge the existence of force, if only to marginalize it. Force is one way that one claims authority, the other is to ride on pre-existing norms.

A force is what causes an event to happen, whether we understand its origin or not. Without forces there can be no causality. The causal chain for change is the following:

$$action \rightarrow force \rightarrow event \rightarrow state - transition \tag{7.13}$$

In other words, a decision to act is followed by the application of a force which results in an event that potentially changes the state of one or more agents.

If an action is applied by agent A to agent A' to force an event that changes A''s state, then we can say that A' has been attacked by A, forced or made to change state involuntarily.

Definition 48 (Force) *An external or environmental influence that all agents are vulnerable to. It is what causes involuntary events to happen.*

The environment itself can impose boundary conditions on the behaviour of an agent, e.g. by putting it in a cage or when passing through a tunnel. This boundary values might also be considered forces. We are used to thinking that time is a force of involuntary change as it seems that we have no way of avoiding the effects of time. However, this we shall return to question this view in the next section.

Definition 49 (Attack) *An attempt to force an agent to cooperate non-voluntarily.*

The creation of an involuntary obligation can thus be considered a form of attack.

Note that it is possible to model forces as agents (just as in the past natural forces were attriuted to "acts of the gods" or to the impact of "particles").

An obligation is different from a force. It is only an assessment of another agent's intentions. In this regard, the use of obligation in security and management systems is somewhat misleading, as there is the implication that an obligation leaves no choice to the agent. This is reflected, for instance, in economics by the two prevailing views of agency: The Theory of State and thence the Law of the State assume the existence of irresistable force. Without this, one has only voluntary cooperation to fall back on. The Theory of the Firm and the market fits more closely the notion of voluntary economic cooperation.

7.6 Laws of behaviour for autonomous agents

We expect to find laws of conservation and change in any theory of behaviour. Intuitively one might easily expect the following:

1. An autonomous agent continues with uniform behaviour, unless it accepts an influence from outside.
2. The observable behaviour of an agent is changed when promising to act on input from an outside source (see section 7.6.1).
3. Every external influence $+b = \langle +\tau, \chi_1 \rangle$ promised by an external agent must be met by an equal and opposite promise $-b = \langle -\tau, \chi_2 \rangle$ in order to effect a change on the agent. If $\chi_1 \neq \chi_2$, then the interaction is of magnitude $\chi_1 \cap \chi_2$.

We shall show that basic laws of this form do indeed apply. In addition, one should expect behavioural properties of any ensemble of agents to be guided by three things:

- The internal properties of the agents themselves.
- The nature of agents' links or bonds (promises).
- The boundary conditions of the environment and location in which the agents evolve.

7.6.1 Behavioural trajectories

To discuss behaviour over time we need to notion of a *trajectory*. This is the path taken by (i.e. the set of intermediate states between the start and the current value of) an agent's observables through a space of states that we may call configuration space. It represents the past or future history of an agent's state transitions. Let $\vec{q}$ be a vector of state information (which might include position, internal registers and other details)[Bur03]. Such a trajectory begins at a certain time t_0 with a certain coordinate value $\vec{q}_0$, known as the *initial conditions*.

The trajectory of a single agent is then a parameterized function $\vec{q}(t, \vec{\sigma})$, for some vector of parameters $\vec{\sigma}$ arriving from an outside source, and we identify the behaviour of an isolated system as the triplet as the determined trajectory:

$$\langle \vec{q}_0, t_0, \hat{O}(\vec{\sigma}) \rangle, \; t > t_0. \tag{7.14}$$

The symbol $\hat{O}(\vec{\sigma})$ is a constant transition matrix or operator which takes $q(t_i)$ to $q(t_{i+1})$ for integer time index i, or alternatively $q(t)$ to $q(t + dt)$ in a differential

form. We can think of this operator as being the generator of time slices, advancing by one time step on each operation; $\hat{O}(\vec{\sigma})$ therefore represents a steady state behaviour and any alteration to this steady state behaviour must come about by a transformation $\hat{O} \to \hat{O}'$, which by the rules of algebraic invariance must have the form $\hat{O}' = T^{\dagger}\hat{O}T$ for some matrix T and dual-transpose representation $\dagger$[1].

In other words, any change in an agent's state (called its behaviour) is generated by

$$\vec{q} \to \vec{q}' = \vec{q} + \delta\vec{q} = \hat{O}(\vec{\sigma})\vec{q} = (1 + \hat{G}(\vec{\sigma}))\vec{q}. \tag{7.15}$$

i.e. $\delta\vec{q} = \hat{G}(\vec{\sigma})\vec{q}$. $\hat{G}(\sigma)$ is called the generator of the transition $\hat{O}$; $\delta\vec{q}$ plays the role of a generalized momentum or 'velocity', so that the dynamics state is represented by the canonical pair $(\vec{q}, \delta\vec{q})$.

We now have a simple transition matrix (or state machine) formalism for describing the steady state behaviour of an agent, which results from keeping its promises through the repeated action of a 'promise keeping operator' $\hat{O}$. An agent whose observable properties do not depend on any external circumstances has *exact* or *rigid* rigid behaviour[ea05a, ea05b]. It is possible if and only if the agent has no use-promises that pertain to its own behaviour ($-b$ for some b), and all other promises $+b'$ are exact promises. In this case the internal change operator $\hat{O}$ cannot depend on any external information.

7.6.2 Outcomes and goals

The notion of a trajectory as a representation of behaviour allows us to be more precise about the meanings of other commonly used terms. We define the *collective behaviour* of several agents simply as the bundle (i.e. direct sum) of trajectories of the ensemble of agents.

An *outcome* (which can equally well refer to the outcome of a promise or of a transition taken to keep a promise) can be described as a single point $q(t_{\rm final})$ of the configuration space reached at some 'final' time, along the trajectory of an agent. In other words it is an identifiable end-point of an agent's behaviour.

A *goal* is then a set of one or more desired or acceptable outcomes within an agent's own state space. In other words a goal is a bounded space-time region that the agent would like its trajectory to intersect (like the bull's-eye of a target). A set of $\{q(t)\}$ possibly over some region $t_{\rm min} \leq t \leq t_{\rm max}$. This definition obeys the principle of autonomy, namely that an agent may only promise its own

[1] This is a linear transformation. It is not certain that all transformations of the operator need be linear, but we make this assumption here for later work to extend. Ref. [Bur04b] shows examples satisfying such linearity, and our results here are only for linear mechanics.

behaviour; however it leaves open the question of how an agent might desire to change its environment (which is outside of its own state space). We come back to this point in section 7.8 since it requires the notion of force.

We wish to point out that a goal cannot be an elementary concept like the subject of a promise, since it requires a feedback loop to achieve, which requires several promises. A goal requires knowledge of the state to be reached and therefore the ability to observe the state and its current state to know when intersection has occurred. As long as the state is internal to the agent we can assume it can simply make these promises itself. However, the situation is much more complex where multiple agents are involved. A collective goal requires all agents to achieve a pre-arranged goal simultaneously. This requires not merely private promises but coordination and hence multiple two-way communication between the agents.

Notice also that the concept of a goal requires the notion of a value-judgement about what is desirable or acceptable. This is easily provided in the promise framework if we always refer to the outcomes of promises, but again it is highly complex where multiple agents are involved.

Finally we should at least mention the notion of non-deterministic states, i.e. macro-states in which a goal is achieved only on average over some interval of time. A promise, after all, lasts for some time and is verified perhaps several times. A promise therefore leads to a distribution of outcomes in general, not merely a single state. One may thus define an *equilibrium* as a goal that is satisfied by a stable distribution over a 'sufficiently persistent interval of time'. As this raises many questions to be answered about the statistical mechanics of agents, we shall defer a full discussion of statistical behaviour for later work.

7.6.3 Changes to steady state

Now, consider how an agent might exhibit behaviour that is based on input from another agent. To see how we might affect a change in this behaviour generated by $\hat{O}$ we need to follow the straightforward rules of matrix transformations. Reactive or adaptive behaviour means that autonomous agents make promises to accept input from an external agent. Thus the operator must be made functionally dependent on the input $\hat{O} \rightarrow \hat{O}(I)$. This requires a promise binding to accept input conditionally on its provision, e.g.:

$$a \xrightarrow{+O(I)/I} a'_{\rm ext} \tag{7.16}$$

$$a \xrightarrow{-I} a_{\rm ext} \tag{7.17}$$

where I represents a promise of input from an external agent, and $O(I)$ represents a promise of some observable output to another external agent which is conditionally a function of the input, and is kept via the operation of $\hat{O}(I)$.

Let $I \to +\Sigma_\tau$ be the body of a promise to change the generator of behaviour $\hat{O}$ from an external agent: i.e. $a_{\rm ext} \xrightarrow{+\Sigma_\tau} a_i$. The agent whose behavioural generator $\hat{G}$ is being altered promises to accept the change with $a_i \xrightarrow{.\Sigma_\tau} a_{\rm ext}$ and we denote a linear realization of the operator which keeps the promise to use this transformation by the external agent simply by Σ_τ so that we have:

$$\hat{G}_\tau \to \hat{G}' = \Sigma_\tau^\dagger \hat{G}_\tau \Sigma_\tau. \tag{7.18}$$

The generator of this transformation matrix can, in the usual way, be written as σ_τ where $\Sigma_\tau = I + \sigma_\tau$, and

$$\delta\hat{G}_\tau = \hat{G}' - \hat{G} = \sigma_\tau^\dagger \hat{G} + \hat{G}\sigma_\tau + \sigma_\tau^\dagger \hat{G}\sigma_\tau \tag{7.19}$$

What can we say about the transformation matrix? In order to satisfy the principle of autonomy it must have the following properties. Let us define a valuation of a promise known as the *outcome* by the notation: $o(a_1 \xrightarrow{b} a_2)$. The outcome returns a value in $[0, 1]$ where 0 means not-kept and 1 means kept. Intermediate values can be used for any purpose, such as statistical compliance. Autonomy requires us to stipulate:

$$\begin{aligned} \Sigma_\tau &\propto o(a \xrightarrow{-\Sigma} a_{\rm ext}) \\ \Sigma_\tau^\dagger &\propto o(a \xrightarrow{+\Sigma} a_{\rm ext}) \end{aligned} \tag{7.20}$$

so that

$$\begin{aligned} \delta G &\to 0 \\ \Sigma_\tau \to 1,\ \Sigma_\tau^\dagger &\to 1 \\ \sigma_\tau \to 0,\ \sigma_\tau^\dagger &\to 0, \end{aligned} \tag{7.21}$$

when one of the binding promises with the external agent is not kept. This means that, unless the promises to deliver an interaction influence are honoured by both parties, then steady state behaviour persists.

The above boundary conditions are the only interpretation of interaction that preserve the requirements of autonomy.

7.6.4 Laws of change

We now state the basic law of causation for behaviour in terms of the autonomous promises of the agents, under the condition of autonomy.

Law 2 (Law of Inertia) *An agent's observable properties hold a constant, deterministic trajectory $\vec{q}(t)$ unless it also promises to use the value of an external source $\vec{\sigma}$ to modify its transition matrix $\hat{O}(\vec{\sigma})$.*

Proof: This follows from eqn. (7.19). Steady state trajectories imply that $\delta G = 0$, which in turn requires that for small changes $\sigma^T = 0$, which implies no promise of type $-\Sigma$. ∎

Put another way, each agent has access only to information promised to it, or already internal to it. A local promise $a_i \xrightarrow{f(\vec{\sigma})} a_j$ that depends on an externally promised parameter σ is clearly a conditional promise $a_i \xrightarrow{f(\vec{\sigma})/\vec{\sigma}} a_j$, where $\vec{\sigma}$ is the value promised by another agent. In order to acquire the value of $\vec{\sigma}$, we require $a_i \xrightarrow{-\vec{\sigma}} a_j$ and a corresponding promise to provide $\vec{\sigma}$ to a_i either from the environment or from another agent. Thus, if an agent does not promise to use any input $\vec{\sigma}$ from another agent, all of its internal variables and transition matrices must be constant.

Note also that by the definitions in [BFa], a conditional promise is only a promise when combined with a use-promise. This fits naturally with the argument in the theorem.

Corollary 1 (A conditional promise is not exact) *By reversing the theorem we see that a conditional promise must, by definition have a residual degree of freedom, which is the value of the dependent condition.*

We can now state the interaction mechanics using the formulations of the previous laws, and in terms of clear statements about state transitions:

Law 3 (Law of interaction) *The acceleration $\delta^2 q$ of an agent's promise trajectory resulting from a promise $a \xrightarrow{O/I} a'$ (i.e. the rate of change of its generalized momentum δq) is proportional to the generalized force $F = \delta\hat{O} = \delta\hat{G}$ promised by an external agent.*

Proof: This now follows trivially from the transformation properties and boundary conditions:

$$\delta_\tau \vec{q} = \hat{G}_\tau \vec{q} \tag{7.22}$$

where $\hat{G}_\tau$ is the matrix valued generator of behaviours of type τ, see eqn. (7.15). Under a change of G

$$\begin{aligned} \delta\vec{q} &= \hat{G}\vec{q} \\ \delta\vec{q}' &= \hat{G}'\vec{q} \end{aligned} \tag{7.23}$$

thus

$$\delta^2\vec{q} = \delta\vec{q}' - \delta\vec{q} = (\hat{G}' - \hat{G})\vec{q} = \delta\hat{G}\,\vec{q}. \tag{7.24}$$

■

Law 4 (Transmitted force - reaction to influence) *The effective transmitted force due to a promise binding between two agents is that which results from the outcome of the body-intersection of equal but opposite ($\pm$) promises between the agents.*

Proof: By the assumption of autonomy, the influence of agent a by $a_{\rm ext}$ is the conjunction of information sent and information accepted: $influence = offer \wedge acceptance$. This has an obvious set theoretic formulation[BFa]. From the rules of promise composition, the binding

$$\begin{aligned} a_{\rm ext} &\xrightarrow{+\langle\tau,\chi_1\rangle} a \\ a &\xrightarrow{-\langle\tau,\chi_2\rangle} a_{\rm ext} \end{aligned} \tag{7.25}$$

has an outcome that satisfies:

$$\begin{aligned} o\left(a_{\rm ext} \xrightarrow{+\langle\tau,\chi_1\rangle} a, a \xrightarrow{-\langle\tau,\chi_2\rangle} a_{\rm ext}\right) = \\ o\left(a_{\rm ext} \xrightarrow{+\langle\tau,\chi_1\cap\chi_2\rangle} a\right) o\left(a \xrightarrow{-\langle\tau,\chi_1\cap\chi_2\rangle} a_{\rm ext}\right), \end{aligned} \tag{7.26}$$

so that the interaction is the intersection of the agents' promises to give and receive the influence. ■

Thus we can say that the trajectory's transformation must have the form:

$$\underbrace{\delta\hat{O}}_{\rm Force} \simeq \underbrace{o(a_{\rm ext} \xrightarrow{+\Sigma} a)}_{\rm field}\,\underbrace{o(a \xrightarrow{-\Sigma^\dagger} a_{\rm ext})}_{\rm charge} \tag{7.27}$$

There is a reassuring correspondence here with the physics of force fields, which is a directly analogous construct, where $\pm$ charge also labels which particles promise to respond to one another's field. Promises appear like fields of influence whose values are sampled by the action of measurement. Promised behaviour is represented by the regular application of operators $\hat{O}$ on a state vector that evolve the state and keep the promise. The outcome is unknown until the act of verification is initiated, somewhat analogous to the quantum theory of matter.

We emphasize that these laws derive directly from the assumptions of autonomy, operational change and transition matrix formulation of the agents. They are therefore beyond dispute and we would expect to find this kind of law in any system of change with similar properties.

7.7 Time and events

Time is generally considered to be involuntary. However, the importance of time is not the same in all problems. Time's relevance to a problem depends on there being interactions that measure its passing. A system in a steady, unchanging state has no knowledge of time at that level of description. Time is therefore not an absolute quantity.

Promise theory is mainly about the analysis of epochs in which promises are essentially fixed. If basic promises change, we enter a new epoch of the system in which basic behaviours change. For a fixed static set of promises, behaviour continues according to the same basic pattern of interactions between agents and environment. This is not a limitation as we shall see in this and the following chapter, but it brings a simple order to the analysis[28].

7.7.1 Time and action (relativity)

It is conventional to think of time as being external to processes: processes are said to take place *in* time. From an observational viewpoint, the opposite is true. Without changes of state, time cannot be observed, and that which cannot be observed can not be claimed to exist for any observer.

7.8 Forces, environment and external goals

By the first law, systems are most predictable when completely isolated from external forces. At the next level, their changes can be predicted when coupling to external forces is weak. The stronger the coupling between agents, the more unknown information enters each agent. This can lead to disordered behaviour which requires more information than is practical or available to understand.

In promise theory all agents begin by default in a state of isolation, impervious to outside influence. It is only through their own promises that they can volunteer to be influenced.

Three questions remain in the discussion: i) how do we explain irresistible forces such as weather, power-failures and other 'acts of god'? ii) How do we model the fact that an agent can affect its environment, e.g. draw graffiti, move an object etc? Finally, iii) how do we model the presence of boundary conditions, or restrictions over which agents have no choice? The concept of force is similar to that of an attack:

Definition 50 (Attack/Force) *An attempt to alter an agent's trajectory without its consent (i.e. in the absence of a use-promise). This is a breach of autonomy.*

Let us consider these briefly for completeness, but defer a full discussion for later work. There are two classical approaches one might take to modelling environmental forces. The first is to think of the environment as simply one or more surrounding agents that distinguish themselves by the magnitude of their influence. The alternative is to treat external objects including the system boundary as being "something else", i.e. some kind of external object that is not an agent. To justify the latter approach we would have to extend the framework of this paper to say what we mean by such an external force and thus we avoid this in the present work. We wish instead to give a very simple view of environmental interaction by treating the environment as a single "super-agent" which promises to allow itself to be changed by any agent and to which all agents have "voluntarily" promised to be influenced. Although this is somewhat artificial[2], it allows us to continue our simply formalism without unnecessary complications.

How can an agent move an object in promise theory? The state of the object needs to be represented in a state space and we must be able to discern its trajectory. If the object is of sufficient importance we can model it as a separate agent that promises to allow itself to be moved by another. Alternatively we can consider all such objects to be mapped into the state space of an environmental super-agent. This super agent can be influenced and can influence agents.

Definition 51 (Leaky agents) *We define a* leaky agent *to be an agent making any promise to receive information from the environment E,* $a_i \xrightarrow{-\mathrm{env}_i} E$.

[2] In fact it is no more artificial than giving certain particles "charge" and defining the notion of a field in physics.

The study of real systems is therefore a study of leaky agents. The environment itself is also leaky in the sense that it can be affected by other agents. This is how we account for stigmergy for example.

With this view, we apply boundary conditions or coupling to the environment by giving every agent a use-promise from this environmental agent to allow some non-specified environmental conditions to be explicitly modelled. The environment agent is assumed to promise its information to all other agents. This is also the way to understand how agents making non-rigid promises can exhibit random behaviour. In order to justify random behaviour we must explain how disordered information enters the agents and selects values from within the bounds of the inexact promises. This is the only mechanism for exhibiting fluctuating behaviour.

By modelling forces using fictitious promises we can use the three laws above to explain all changes in a system in a common framework. Regardless of whether one finds this to one's taste, it is a rather practical step for simple modelling. We add finally that the concept of a goal might now be extended to allow agents to desire outcomes about states in the environment, not only in their own state space. This is reasonable for any agent as long as it has a use-promise from another agent to accept changes of state. However, a goal is still not an elementary concept that can be the subject of a promise – it is an outcome that might emerge from the behaviour.

7.9 Emergent behaviour and goals

When is behaviour designed and when does it emerge? Promises are designed but outcomes emerge. Leaky agents especially can be influenced by environment and we cannot completely determine their trajectories. We speak of emergence when we identify behaviour that appears organized, but where no perceptible promises to account for this have been made.

Many authors have fallen into the trap of using the terminology of goals to describe emergence – goals which the parts of the system are incapable of knowing individually. This is a superfluous explanation which likely emerges from the fictitious belief that programming determines real world behaviour. We have shown that this is not the whole story and now offer a simple explanation for emergent organization.

To understand emergence we must look to the spectrum of observable outcomes of agents' promises. Inexact promises allow for unpredictability and the question is to understand whether organized behaviour is likely, in spite of not being an agreed cooperative goal of the agents. We have proposed that promises

must be inexact to allow for the possibility of unpredictable behaviour[BF07] and that the following simple definition of emergent behaviour is plausible and captures the popular views in the literature.

Definition 52 (Emergent behaviour) *Emergent behaviour is the set of trajectories belonging to leaky agents exhibiting non-rigid, collective behaviour that is observationally indistinguishable from organized behaviour.*

The important issue here is observational indistinguishability. It is the end observer who looks for 'meaning' (i.e. a goal) in the organized outcomes; the actual promises made by the agents could in fact be anything that allows the observed outcome to arise. In other words an outcome 'emerges' simply because it arises.

There are many mysterious definitions of emergence in the literature but emergent behaviour can be understood easily by looking for any promises that enable the observed outcome and using algebraic reduction to account for behaviour as in section 8.4.5, keeping firmly in mind the notion of observational indistinguishability. After all, if emergent behaviour is real, it should ultimately be measurable by the same standards as any other kind of behaviour.

The key to emergence then is that the residual freedoms in the agent promises (i.e. that are not constrained exactly), are selected from by interaction with the environment, resulting in patterns of behaviour that are unexpected, but which nevertheless lie within the bounds of the promises given.

An example of emergent behaviour often cited is the idea of a *swarm*. Many definitions of swarms have been offered[BDT99, KE01, Woo02, CD98, ADL05, S00, Hey07, Hol98]. Ours is simply as follows:

Definition 53 (Emergent group or Swarm) *A collection of leaky agents that may be seen by any external observer as exhibiting undifferentiated, collective behaviour.*

7.10 Causation, Equilibria and Newcomb's Paradox

Which came first, the chicken or the egg? Causation is not always an appropriate description of phenomena in which there is no arrow of time. Such phenomena are known in the natural sciences as 'equilibria'.

Equilibrium can be a statistical phenomenon, so that there are many changes, all happening in a forward temporal direction, but the outcomes of these changes are measured in such a way that there is change both forwards as backwards, and thus the average affect is zero.

Equilibrium can also be 'exact', i.e. equal and opposite changes that cancel instantaneously. This is true in 'static equilibrium' such as a person not falling through the floor, since the weight of the person is cancelled by the force of the floor pushing up (expressed by Newton's third law). It can also be a dynamical cancellation, such as in the interference of waves where peaks and troughs cancel (the principle used in sound-cancelling headphones), or someone running backwards down an escalator.

7.10.1 Prediction and Newcomb's Paradox

Consider the following chain of events. An agent whom we call the 'player' is asked to make a rational choice about whether to lift one of two cups in front of him. By choosing either the left or the right cup, he can claim a prize. The rational choice is (by definition) that choice which maximizes his reward.

By analyzing the probabilities (as one would in game theory), it comes to be known (by whatever means) that that the probable reward for choosing the left hand cup is greater, so (rationally) that is the best strategy. As long as the information is trustworthy, this is the best solution and there is no argument. Game theory selects the best strategy is 'choose the left cup'.

The suggestion of paradox occurs when one introduces another agent that claims to be able to predict the future exactly – not merely statistically. This second agent, the 'predictor' makes a prediction about what the player will do. This prediction is assumed to be correct. The game tree is depicted in fig. 7.4, time is measured downward.

The figure depicts a traditional view of a game, drawn as a tree because there is an implied causation, or arrow of time. Information moves in the direction of the implied arrows (downward). But this is not the case if the Predictor has certain knowledge of the future.

From the diagram, as drawn, we seem to imply that there are 4 possible pathways:

L	R	0
L	L	1
R	R	50
R	L	100

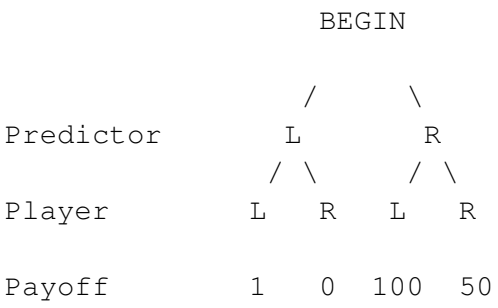


Fig. 7.4. Newcomb's Paradox Scenario.

Without the prediction, "L" is always the "dominant strategy" in game theory parlance. But with the predictor, there is no dominant strategy.

The paradox is supposedly this: if we ignore the predictor then L is the *dominant strategy*: 1 is better than 0, and 100 is better than 50. However, the highest payoff 100, which can only be reached by ignoring the advice of the predictor. Thus a rational agent would have to defy certain knowledge of the future to get this reward – or put provocatively: the knowledge of the future forces the rational agent to make an irrational decision.

It is easy to see that the source of the so-called paradox is careless bookkeeping of assumptions. Either a predictor knows the future or it does not. Either it is correct, or it only guesses. As soon as we add the genuine predictor that knows the future, there is a new *absolute constraint* on the allowed transitions: namely that the predictor does indeed speak the truth about the choice made. This implies only two possiblities L, L or R, R. With this constraint, there is no paradox. The path R, L is simply fictitious, because it can never happen from the assumption of prescience.

Is this realistic? Well, the paradox scenario does not claim to be realistic of course. It is meant to provoke criticism of the notion of a rational agent. There are no perfect predictors of the future, naturally. Suppose then that the predictor does not have certain knowledge (only probable knowledge, as one might expect of a good weather forecast), then we immediately fall back to a classic gambling problem about managing expectations.

The confusion happens perhaps because game theory has no notion of 'absolute constraint' that forces us to select either L, L or R, R, so sneaking a constraint in the back door brings confusion. Moreover, since there is then no dominant strategy L or R for a fake-predictor, the scenario simply reduces to a problem of *incomplete information*. The problem is 'under-constrained', and it

does not have a unique solution until some other inputs arrives to further constrain things.

Newcomb's paradox is thus a confusion about the meaning of freedom and constraint. It clothes itself in the garb of game theory (see chapter 15), where so-called rational agents that are supposed to make choices about the future based on their expected rewards. The paradox however is a muddle about what kind of rationality one is allowed to expect of a so-called rational agent, constrained to act in a particular way by forces beyond its control. In other words, it is about a lack of freedom to choose.

7.10.2 Promise equilibrium in the paradox scenario

Let us consider a very simple model of the scenario using promises. The actor promises to act and the predictor promises to predict. For the sake of argument, let us assume that they make these promises to one another.

We can construct a proof that rationality is impossible under these constraints...

Notes

[18]The ability to model human motivations without tearing down the formalism of rigour is essential to reconcile physical and social sciences. Many physical scientists are rightly suspicious of social sciences and humanities when they abandon the forms of clear expression with the explanation that humans cannot be pinned down in that way. We maintain that necessary and sufficient description of characters must form the basis for any description of the world.

[19]Physicists know that these 'laws' are not in fact observed with one hundred percent reliability, due to experimental errors and even environmental or quantum uncertainties. For a full discussion, we must discuss the problem of measurement in chapter 8

[20]For example, we can only observe colour because our brains are able to classify optical information at each point into mixtures of separately assessable states: red, green and blue. A ruler with millimetre markings can only be created because matter can exist in different coordinate states, labelled by distance from an origin. If one were not able to distinguish red from green, observers would be colour-blind. If one were not able to distinguish distance separation, as when looking along the axis of separation (as with the stars in the sky), then we would not be able to say which object was closer. If no change in an environment (like the different states of position of a clock's hands) is registered, we cannot even measure the passing of time.

[21]One can of course model even this elementary requirement with promises. If the labels are each considered to be agents and each agent promises to depend on the precedence of another, then they will order themselves naturally.

[22]Such stepping operators are well known in group theory, e.g. for Lie algebras, where they are related to the roots and weights of the Lie algebra.

[23]In quantum theory a "particle" is simply such a change-counter.

[24]In physics, equilibria play a dual role. They are 'macrostates' in which there are no observed changes, and yet they also form the the latent generators of time ordered processes. The so called

effective action, related to the 'free energy' in thermodynamics, is the sum of all the different ways that nothing (on average) can happen in the system[Rei65, Abb92, Bur02]. This generates a lot of symmetries. By breaking any one of these symmetries, something will happen. Thus the effective action equilibrium state is the starting point for all the ways a system can change. The variation of an equilibrium subject to time-ordered boundary conditions breaks time-reversal symmetry and leads to equations of motion in a single time direction. We can use the same principle in promises.

[25]A cycle in this instance is somewhat similar to the idea of a plane wave in physics. It is a basic Fourier process that represents a countable 'quantum' unit. In quantum physics, this is often referred to misleadingly as a particle.

[26]In mathematics the of difference equations, the solution has two parts: a 'particular integral' which describes the transient response to the initial 'kickstart' condition, and the 'continuous integral' or complementary function that describes the steady state behaviour, once all memory of the transient response has subsided. This is directly analogous to the situation here.

[27]This is completely analogous to the way physical law is expressed in terms of differential or operator equations that are reversible, where in reality only one of the directions is observed. The point is that time itself is not unique – it is defined by an ordered sequence of change. That is why is it possible to construct both 'advanced' and 'retarded' viewpoints for matching phenomena to boundary conditions at different points along a generated sequence.

[28]We are just making the same kind of approximation used in calculus of small differentials, in which the background is assumed constant over each differential. If the intervals are small enough compared to the rate at which the promises are changing, then the approximation is exact.

8
Assessment and Measurement

So far we have only discussed the mechanics of promises as a way of formalizing intent. We have not considered, for example, how to vetify whether a promise is actually kept or not. Ultimately, promises are an approach to expressing knowledge about a model of intention, making the best of the information one has from a subjective, black-box view of the world.

Since, promises span every possible kind of behaviour, we would like to choose a form of assessment that is generic, reducing every assessment to a binary determination of whether a reference promise has been 'kept' or 'not kept'. This is simplistic, of course, but it follows in the Western tradition of mechanization.

Assessment of promises is an inherently subjective matter. Different agents might arrive at very different conclusions based on their own circumstances. Assessments must therefore respect the relativity of agent perspectives. This is to be expected.

A number of things can complicate a promise assessment, such as cooperative behaviour between the agents, whereby one agent is only willing to assess a promise as 'kept' if it receives the 'ok' from another agent.

The desire to more objectively and accurately assess the outcome of promises motivates the *observation* of agents. In doing so, we shall be immediately forced to confront the subjectivity of such the assessments. This, in turn, will bring us right to the heart of modern descriptions of the world that embrace indeterminism.

8.1 Defining assessment

Let us progress from treating assessment as a primitive, to defining promise assessment in terms of observation. We shall be careful to avoid the more tempting phrase 'verification' of promises, as this suggests a level of objective verification which is not attainable. Our notion of an assessment is more generic and less quantitative than a verification. It is both subjective and not *a priori* linked to observation, but we shall explore the larger implications of observation in this chapter.

8.1.1 Some assumptions

Every promise is assessable. We assume that from the time a promise is published, in its lifecycle (see section 7.1), a process of assessment begins in all agents that are aware of this promise. The assessment may or may not be a conscious attempt to assess the outcome, it may or may not be a rational attempt, but some result based on the subjective ruminations of each agent forms an on-going process of revised estimation. Looking at the promise life-cycle:

$$\begin{aligned} Promiser: \quad & proposed \\ & \downarrow \\ & issued \\ & \downarrow \\ & (unknown + kept + not\ kept)^* \\ & \downarrow \\ & broken + withdrawn + expired + invalidated \end{aligned} \tag{8.1}$$

the repeated process of assessment may be identified as the starred parenthesis $()^*$. This parenthesis contains the possible assessments made on each reassessment of the promise (as many times as desired until the expiry of the promise, etc.)

We may assume that each assessment lasts an arbitrary time Δt_i for agent A_i. These intervals are the choice of the agents themselves, and reflect the trust the agent has in the stability of the assessments. In other words, an agent who suspects that a promise status might change suddenly, might monitor the outcomes and reassess the promise very often. Similarly, a very fickle or unstable agent might do the same. A very trusting agent might believe it sufficient to make a single assessment that is never revisited.

8.1.2 Labelling evidence

To incorporate observation into our theory, it will be important to appreciate how observations are subject to uncertainties and subjectivities. Regardless of whether an agent knows about a promise (because it is in its scope), and has access to some information about it, observers with different capabilities, viewpoints, or frames of reference will naturally assess outcomes differently, just as different experimenters will obtain different results and conclusions when examining the world in independent trials. Furthermore, if observations are subject to the capabilities and limitations of the agent making the assessment, then each assessment of a promise should be labelled by the agent making the assessment. That agent then becomes responsible for it; indeed, the assessment is effectively a promise made by that agent.

More labels will be necessary to explain the context in which the observation was made. Time and perspective must play a role, for instance, as each assessment might only have a finite lifetime during which is is considered valid: if an agent looks only once, the assessment might be negative, but another observation a few moments later could change that assessment. Finally, the times at which the measurements supporting the assessment were taken need to be documented to clarify the supporting evidence.

Humans often make assessments based on no evidence at all, or at least indirect evidence that builds trust. One might say: 'when I release the hammer, it falls – this has always happened before, I see no reason why it would not happen now'. In a familiar context, this assessment might be reliable; in outer space, it would be false. Whether this kind of inference is useful or not depends on some assessment of its relevance.

8.1.3 A definition of promise assessment

Definition 54 (Assessment) *An assessent of a promise π may be any specified function that asserts whether the promise was kept or not kept. It is written $A_a(\pi; t_i, t_j; I)$, where a is the agent making the assessment, π is the promise being assessed, t_i is the initial time and t_f is the final time of the assessment interval, and I is a set of impressions on which the judgement is based.*

Note that the assessment function itself makes a promise, namely one to supply its determination of the outcome, thus it is not strictly a new kind of object in the theory of promises. The impressions I are to be thought of as points of data, either invented, received, or specifically measured about the promise.

These might include evidence fom measurements M, or even hearsay from other agents, in the general case.

Traditionally experimental science has viewed all empirical evidence as being 'true' subject to some uncertainty, so it could seem perverse to suggest it might be acceptable to invent an assessment, but it is also known that evidence can be invented, distorted or misinterpreted (accidentally or purposely) by agents human or mechanical in any experiment, regardless of the desirability of this, so a serious theory of assessments in relation to promises must include such issues[29]. The latter view, in fact, moves us closer to theories like quantum physics where one is forced to take into account the effects of incomplete information and the lack of complete knowledge of the apparatus itself in the result. Ultimately, every assessment is based on a *trust* of the supposed facts. Here we make that explicit[30].

8.1.4 Auxiliary definitions of assessment

It is useful to define two functions of a promise that indicate assessments of outcome. These correspond roughly to the two interpretations of statistical data: namely a belief (Bayesian) interpretation, and a frequentist or evidential interpretation.

Definition 55 (Belief in outcome $\beta(\pi, t_i, t_f, I)$) *An assessment made without observation $\beta : \pi, t_i, t_f \to [0, 1]$ of the likelihood that a promise π will be kept within a stated time frame $t_f - t_i$.*

Definition 56 (Evidence of outcome $\phi(\pi, t_i, t_f, E)$) *An assessment made with partial information $\phi : \pi, t_i, t_f \to [0, 1]$ of the likelihood that a promise π was kept within a stated time frame $t_f - t_i$.*

8.1.5 Inferred promises: hypotheses

Sometimes systems appear to an observer to act as though they keep certain promises, when in fact no such promises have been made. The observer might simply be out of the scope of the promise, but can still observe its repercussions; or, in fact, no such promise might have been made at all. Does this matter? From the perspective of the observer, it makes no difference whether a promise has actually been made or not as long as the agent appears to be behaving as though one has been made. It is entirely within the observer's rights to postulate a model for the behaviour in terms of hypothetical promises.

One example of this is the way science approaches the 'laws' of nature: it appears that the world makes certain behavioural promises, which we can codify into laws. In truth, of course, no such laws have been passed by any legal entity. Nature seems to keep these promises, but no such promises are evident or published by nature in a form that allows us to say that they have been made.

Definition 57 (Inferred promise) *Starting from an assessment* $A(\pi_{\text{unknown}}; t_i, t_f; I)$ *of impressions* I*, over an interval of time* $t_f - t_i$*, one may infer the existence of one or more promises fitting* π_{unknown} *that fit the assessment.*

An observer cannot know whether its hypothesis is correct, it can only accumulate evidence to support the hypothesis.

As an example: suppose a vending machine agent is observed to give out a chocolate bar if it receives a coin of a certain weight. Since most coins have standard weights and sizes, a thief who was not in possession of this knowledge might hypothesize that the machine actually promised to release a chocolate bar on receiving an object of a certain size. Without further evidence or information, the thief is not able to distinguish between a promise to accept a certain weight and a promise to accept a certain size, and so he might then attempt to feed objects of the right size into the machine to obtain the cholocate bar. Based on new evidence, he might alter his hypothesis to consider weight. In truth, both hypotheses might be wrong. The machine might in fact internally promise to analyze the metal composition of the coin along with its size and other features.

The point of this example is that assessments are difficult to make, if one is trying to reverse-engineer the promise that led to an observation, because there is no unique one-to-one correlation between intent and what can be observed.

8.2 Boundary conditions for assessments

Because assessment is based on changes of state, within a usually limited time frame, there are several possible models for assessment. The choices reflect which points of 'certain' knowledge the observer has chosen to trust the most. These points are referred to as boundary values, or initial conditions (see fig. 8.1). We use the standard nomenclature for boundary conditions from potential theory[Bur02]:

- Retarded assessment: One starts from an initial state; events occur or not according to the promises of the system, and an assessment of the final state is made at a later time.

- Advanced assessment: Working backwards, one starts with a knowledge of the outcome state, and infers from the known promise, an assessment or inference of the initial state.

An assessment usually starts with an existing promise, i.e. with expectations of some event in the beginning of an experiment. Later, there is a state in which the assessment has been made and the result has been obtained. In between, there are possibly events. There are incoming states and outgoing states, and the transition function between them represents an assessment of the belief in a transition from one to the other[31]. In Dirac notation, one writes this in the semi-symmetric form:

$$A(\pi; t_i n, t_o ut, M) = \langle \text{out}|M|\text{in}\rangle. \tag{8.2}$$

This causal chain is an essential part of the assessment.

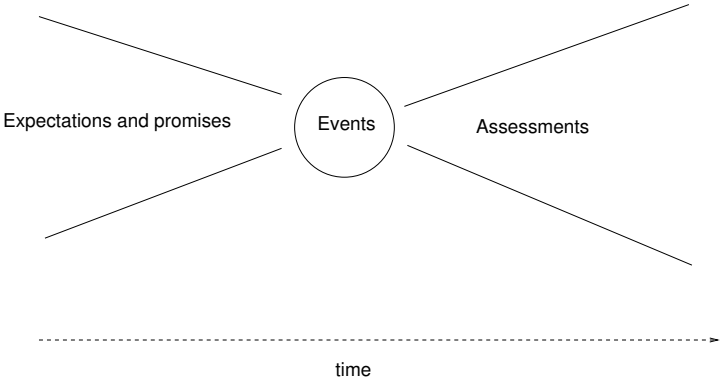


Fig. 8.1. One possible model of causal boundary conditions on the assessment process. Here the promise comes first, something happens, and the result is assessed.

8.2.1 Linear assessment functions and valuation

An assessment function may be called linear if it satisfies the relation:

$$A_a(\pi; t_i, t_f; \sum_i a_i M_i) = \sum_i a_i A_a(\pi; t_i, t_f; M_i). \tag{8.3}$$

This form of assessment function can be used as a notion of value for accounting purposes, thus it allows us to associate payoff, or utility to promises in a manner consistent with game theory and economics. In particular, we note that a linear

combination of sample measurements $\langle M \rangle = \sum_i \delta^i M_i$, where δ^i means raised to the power i, and $\delta < 1$, represents a discounting of value over time, as is common in economic valuations.

Although linear functions are especially important, they are not the only assessments. A form of some importance is the quadratic invariant that represents the Euclidean norm:

$$\mathcal{A} = \sqrt{A_1^2(\pi; t_i, t_f; M_1) + A_2^2(\pi; t_i, t_f; M_2) + \ldots}. \tag{8.4}$$

This is used in the theory of errors and uncertainties to compute a fair metric distance for independently assessed uncertainties. It corresponds to the Pythagoras rule.

8.3 Conditions for observation

An assessement may or may not be based on the observation of an agent. It can be entirely imagined, or obtained indirectly by reasoning. It is nevertheless desirable to obtain the most accurate information about an agent's state and this can only be obtained through direct observation.

8.3.1 The observation interaction

Consider a set of observables $O_\alpha, (\alpha = 1, 2, \ldots)$, i.e. variables that take numerical values. A successful measurement of these observables is an interaction between two parties, requiring mutual promises.

In the voluntary cooperation model of promises, the amount of information transmitted is the overlap between the 'willingness to transmit' and the 'willingness to receive' of the agents involved. The 'willingness to receive' acts as a filter on what will be accepted by the receiver. It is thus a basic process of information transfer[SW49, CT91]. Irrespective of notions of freewill, our model of promises serves a role in making clear the necessary and sufficient conditions for transmission of data. From there, it is possible to address the limitations we must expect to observability.

The argument begins along parallel lines to the concept of mutual information in communications theory[SW49]. A measurement is the information that results from that part of what the sender reveals that is collected by the receiver. This requires two promises: a promise to reveal or send information by the agent being measured a_s and a promise to receive that information from the

agent making the measurement a_r.

$$a_s \xrightarrow{+data} a_r \tag{8.5}$$

$$a_r \xrightarrow{-U(data)} a_s \tag{8.6}$$

Schematically the resulting promise is the overlap, or intersection between these constraints (see section 7.4).

$$\text{Measurement} = \text{Receiver reception} \cap \text{Sender data transmission} \tag{8.7}$$

A receiver might be unable to receive information from the sender or even unwilling to receive information from the sender, in the case of filters. Thus the overlap operator $\cap$ covers both eventualities.

$$\text{Measurement} = \text{Filteredsenses} \cap \text{Observables} \tag{8.8}$$

This makes it clear that there could be limitations of perception and capability involved in transferring accurate information from the item being observed a_s to the measurement apparatus a_r.

$$\text{Measurement} = (\text{Capability} \cap \text{Acceptance}) \cap (\text{Data} \cap \text{Revelation}) \tag{8.9}$$

Since the number of observables is generally greater than one, they form a vector $\vec{O}$ with components $O_\beta, (\beta = 1, \ldots, m)$. Similarly, the measurements successfully received form a vector $\vec{M}$ with components $M_\alpha, (\alpha = 1, \ldots n)$, and $n \leq m$. The overlap function Ω is thus a matrix with components $\Omega_{\alpha\beta}$, giving the overlap relation as:

$$M_\alpha = \Omega_{\alpha\beta} O_\beta \tag{8.10}$$

The number of measurements acquired is necessarily less than or equal to the number of observables available in general, so the matrix Ω is not necessarily a square matrix.

For the purposes of assessment, an agent needs to turn a set of vectors M, measured at one or more sample times, into a number between 0 and 1 that forms its estimate of the extent to which the promise was kept.

8.3.2 Interference from environment and noise

In many if not most cases an agent that is probing another is not able to promise to limit its reception to receive only what is transmitted by the other agent. Other signals from the environment will get mixed in with what is made available by a_s.

Formally, this can be seen as the mixing of two independent sources: the observables $\vec{O}$ and the environment $\vec{E}$, such that:

$$M_\alpha = \Omega_{\alpha\beta\gamma} O_\beta E_\gamma \tag{8.11}$$

The change in $\vec{M}$ as a result of this

$$\Delta M_\alpha = \Omega_{\alpha\beta\gamma} O_\beta E_\gamma - \Omega_{\alpha\beta} O_\beta \tag{8.12}$$

may be viewed as noise. One often calls such noise random if it is generated by a process about which we have insufficient information to predict its pattern.

Consider an example. Let a_s be an distant star, 20 light years away, and a_r be a camera attached to a telescope, with a collection of colour wavelength filters available.

The star a_s makes available light from its surface, but does not reveal information from below its surface. The telescope a_r receives only light that was sent 20 years ago, and does not receive all wavelengths of light made available by a_s due to its physical limiations and its voluntarily chosen colour wavelength filters.

Along the journey, gases and dust absorb parts of the signal and re-emit additional data that gets mixed into the measurement channel. Some of this is constant, and some changes randomly in time.

The final measurements received between the initial and final assessment times is thus the original signal from the surface of the star mixed and distorted by interstellar gases, and filtered by the telescope.

8.3.3 Interface groups

Every assessment requires the interaction of two agents: the agent being assessed and the agent making the assessment. For each assessment that the latter intends to make, the agent being assessed must promise to cooperate, or conversely for each promised behaviour, the observer must promise to observe it. This means that a pair of promises $\pm b(\pi)$ is required for each assessment, to allow knowledge transfer.

This pair of promises forms an 'interface group'[BP06].

8.3.4 Consequences of agent autonomy for observation

The autonomy of agents in Promise Theory serves as a model for many of the real limitations in observation.

1. Nothing the observed agent does not intend to reveal can be observed directly.

2. The observer can misinterpret information in a number of ways:

   (a) The states that record information inside the observed agent do not match the states the observer has internally, so the fidelity of transmission must be compromised.

   (b) The transmission is interfered with en route.

   (c) There is no common standard of meaning for the information transmitted between agents.

In experiments, researchers often 'attack' systems by force to overcome the limitations of what an agent can receive, e.g. using instruments during autopsies, sending probes to breach the outer shells with X-rays or sonic waves. Rather than viewing this forced approach as breaking the promise model, it is more useful to view it as probing a new level of promises about the internal structure of the black-box agent. This keeps us within a single framework.

The ability to extract more information assumes that the probe forced upon the agent has sufficient resolution to discern additional information however.

8.4 Measurement

Measurement and observation are central to the discussion of system behaviour. The concepts of calibration and coordination must also feature. Promises define the only observables in promise theory. No knowledge is predictably exchanged without it being promised either explicitely or implicitly, so there is no measurable certainty without both promises to be observable and to observe in a binding relationship.

The location and nature of an agent can only be observed if the agent promises to make its position and attributes visible. This is also the case for its interactions with an environment. The environment itself must promise to make its attributes available to agents – this is usually a trivial matter, since the environment is generally observable by all, depending on their sensing capabilties. Although this approach to the problem is clearly a modelling device, the description has the advantage of making explicit all assumptions about individuals and their interactions, including observations of environment and boundary conditions etc.

8.4.1 Algebra of measurement

The need for outcomes to be assessed by a single agent suggests a basic algebra of measurement because it implies that all comparisons must be made through a single adjudicator. Indeed this property explains how certain agents attain a privileged position in an ensemble, by having access to information and using a single scale of measurement for coordination of all the information from observed agents.

We use the shorthand notation $C(b)$ for a pattern of promises that leads to the effective promise "exhibit the same as". The coordination promise is used for this (see fig. 8.2). The reduction rule for coordination promises for the case in

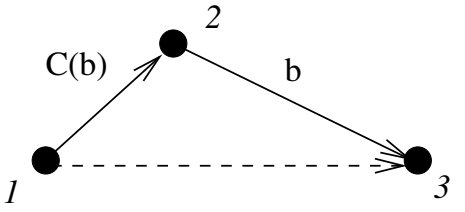


Fig. 8.2. Serial composition of a promise and a coordination promise. The dashed arrow is implied by the $C(b)$ promise.

which A_1 promises A_2 that it will coordinate on the matter of b, given that A_2 promises A_3 b follows. The symbol " , " is used to signify the composition of these promises.

$$\underbrace{A_1 \overset{C(b)}{\to} A_2}_{\text{'Coordinate with'}} \quad , \quad \underbrace{A_2 \overset{b}{\to} A_3}_{\text{Promise}} \quad \Rightarrow \quad A_1 \overset{b}{\to} A_3 \; . \tag{8.13}$$

The coordination promise is transitive.

$$A_1 \xrightarrow{C(b)} A_2 \; , \; A_2 \xrightarrow{C(b)} A_3 \Rightarrow A_1 \xrightarrow{C(b)} A_3. \tag{8.14}$$

We use this below in the identification of observable properties, since it implies a basis for A_3 to compare A_1 and A_2.

Measurement may be thought of as observation with commonly established standards of representation.

8.4.2 Necessary and sufficient conditions for measurement

The promised outcome must be in the scope of the observer. There must be an interface group.

Common standard of knowledge is not required. An observer can form a consistent model of the observed without actually knowing its internal details.

Suppose a promise has not been made to offer insight into the internal promises of an agent. If an observer sees changes in the agent that follow sufficient structure and predictability, the observer should be able to infer the existence of hidden promises. This might be called a speculation or even a model.

Trust or confidence in the information received is another axiom for measurement.

What if the data sent are systematically wrong, or subject to interference?

8.4.3 Indistinguishability – equivalence of outcomes

If two promise bundles have the same potential outcomes (according to some agreed form of verification v), we can say that they are functionally equivalent.

$$A(a_1 \xrightarrow{S} a_2) \stackrel{v}{=} A\left(\left\{ \begin{array}{c} a_1 \xrightarrow{S|X} a_2 \\ a_1 \xrightarrow{\pm X} a_2 \end{array} \right)\right. \tag{8.15}$$

Two promises can further be considered stochastically equivalent if they have the same outcome frequencies over multiple trials, according to some agreed measurement process m (i.e. if the result of the promises can be the same) and the numerical probability of the outcome is equal.

$$\phi(a_1 \xrightarrow{S} a_2) \stackrel{m}{=} \phi\left(\left\{ \begin{array}{c} a_1 \xrightarrow{S|X} a_2 \\ a_1 \xrightarrow{\pm X} a_2 \end{array} \right)\right. \tag{8.16}$$

The important thing about probabilistic compliance with promises is that they wash out the details of how the promise was kept. The probability of two

It does not make sense to say that two promsies of different types are equivalent, since the outcomes would be different even if the probability of their being kept were coincidentally the same.

8.4.4 Distinguishability of agents

It follows from the discussion surrounding the third law of action that the outcome perceived by agents a_1 and a_2 in their observation of a third agent a_3 need not agree. Their promises to receive promised data could be different or even incorrectly calibrated. This is not the only reason why perceived values might differ, but it is a sufficient one. It follows that each agent has an independent

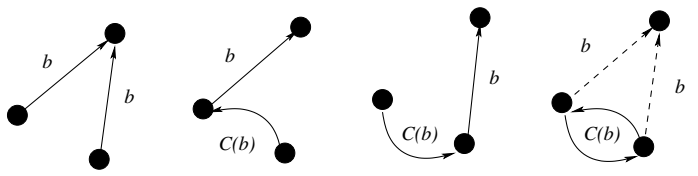


Fig. 8.3. Observation indistinguishability implies an equivalence.

estimation of values observed. How then can any third agent determine whether a pair of agents has the same behaviour? To do this measurement must be with respect to an arbitrary but *singular* observer that can make relative comparisons according to its own subjective apparati.

The coordination rewriting rule can be applied both forwards and backwards. Since an agent that observes behaviour b has no knowledge of what might lie behind it, it cannot tell the difference between the scenarios in fig 8.3.

Since agents cannot distinguish between these cases by observation alone, they are entitled to consider the situations equivalent.

Definition 58 (Equivalence under observation) *A constellation of two or more agents that promise identical observables with equivalent behaviour to a given agent are considered equivalent under observation if the observing agent cannot distinguish the average behaviour of the agent with respect to the promised observable.*

Note that equivalence could be exact at any moment in time, perfectly synchronized changes, or it could be defined as an equivalence of averages over a sampling interval (e.g. a suitable time scale). This must be specified when reasoning about the promises.

The notion of equivalent is not entirely clear however.

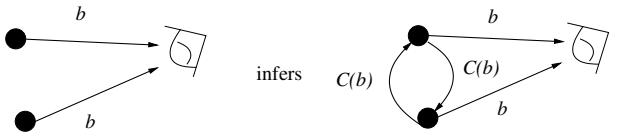
- Equivalent means identical.
- Equivalent only statistically, over a given sample scale, with quantifiable bounds.

These judgements depend clearly on the abilities of the observer to discern and distinguish behaviour, thus two different agents' findings should cannot be compared without prior calibration.

8.4.5 Rewriting for cooperative behaviour

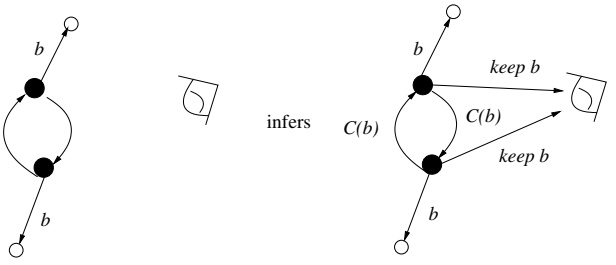
Based on the equivalences shown in fig. 8.3 a number of rewriting rules can be formulated expressing equivalences under our observation as modellers with complete information about all agents (having a globally privileged insight that no single agent has). We shall describe them pictorially here to avoid unnecessary formalism. From the symmetry of indistinguishable agents, the implicit promises between the agents must appear in both directions.

1. *Inferring implicit coordination.* Two agents that behave in the same way to a real or fictitious external observer over some defined timescale can be assumed to be coordinated, and we may introduce symmetrical coordination promises.



2. A corollary to this is that when all agents in an ensemble make identical promises to each other in a complete graph, we can add mutual coordination promises between all pairs. This is easily justified as it reflects the observation that when a number of agents behaves similarly with no labels that otherwise distinguish special roles, an external observer can only say that all of the agents are behaving in a coordinated way. Thus the observer sees a coordinated group for all intents and purposes, although it was not formally agreed by the agents.

3. *Inferring observational indistinguishability.* Any two agents that mutually coordinate their behaviour may be considered to behave analogously over the sampling interval to a hypothetical external observer. This can be formulated by introducing a fictitious promise to the fictitious observer.



We have argued that these rules are the basis for understanding swarm behaviour[BF07].

Coordination of agents could of course be arranged by having each agent subordinate itself to the instructions of a third party, but in this case one must postulate the existence of an additional *real* agent which does not seem justified directly. However, by combining the rules above one can postulate the existence of such an agent entirely on the basis of equivalences, which is preferable.

8.5 Entangled promises

Notes

[29]According to our definitions, scientists regularly fail to keep promises, or even lie about what they know, when promising conclusions about which there is complete certainty. While few scientists would like to see themselves in this picture, the semantics are correct.

[30]In natural science that trust in experimental facts is estimated using the theory of errors, which is based largely on a model of Gaussian random noise. In more complex situations where experiments can be less well controlled, there is even greater uncertainty to deal with.

[31]The role of an assessment is analogous to the S-matrix, or scattering matrix in quantum field theory.

9
Trust and Promise Keeping

I don't trust him. We're friends.
–Bertolt Brecht

The decision to trust someone is a policy decision. Although the decision can be made *ad hoc*, our common understanding of trust is that it is based on a gathering of experience, i.e. a process of learning about the behaviour and reputation of someone in a variety of scenarios. Our particular policy might weight certain sources and behaviours more heavily than others and no one can tell us what is the right thing to do. Hence trust is intimately connected with personal autonomy.

In this chapter, we wish to define trust in the spirit of this personal autonomy, by basing it directly on the concept of how reliably a promise is kept. A promise is also an autonomously made declaration of behaviour, that is highly individual, moreover it carries with it the notion of a theme (what the promise is about)[BB08]. By combining promises with reliability, we thus have a natural definition of trust that satisfies well-understood rules for revising both the logical aspects of policy and the statistical observations made about agents' behaviours. We show that this viewpoint satisfies the desirable properties for use in computer security schemes.

9.1 What is trust?

The concept of trust is both well known and widely used in all kinds of human interactions. However one chooses to interpret the tantalizing quotation above, it should indicate that trust is a subjective and highly non-trivial issue.

Trust is something that humans hold both for one another or sometimes for inanimate objects ("I trust my computer to give the right answer"). In computer

systems, the concept of trust is especially used in connection with security. In risk analysis one considers a secure system to be one in which every possible risk has either been eliminated or accepted as a matter of policy. Trust is therefore linked to the concept of policy in a fundamental way.

Trust is also discussed in the case of network security protocols, for instance, in the case where keys are exchanged. The classic dilemma of key distribution is that there is often a high level of uncertainty in knowing the true originator of a secure identifier (cryptographic key). One therefore hopes for the best and, beyond a certain threshold of evidence "trusts" the assumption of ownership. Several protocols claim to manage such trust issues, but what does this really mean?

In spite of the reverence in which the concept is held, there is no widely accepted technical definition of trust. This has long be a hindrance to the discussion and understanding of the concept. The Wikepedia defines: "Trust is the belief in the good character of one party, they are believed to seek to fulfil policies, ethical codes, law and their previous promises." In this chapter, we address the deficiencies of discussions of trust by introducing a meta-model for understanding trust. Our model can be used to explain and describe common trust models like "trusted third parties" and the "web of trust".

9.2 Trust and autonomous promises

Trust is an evaluation that can only be made by an individual. No one can force someone to trust someone else in a given situation. This basic fact tells us something important about how trust should be defined.

Recently, one of us has introduced a description of autonomous behaviour in which individual agents are entirely responsible for their own decisions[Bur05, BFa, BFb, BF06]. Promise theory is a graphical model of policy. The basic responsibility of an agent to be true to its own assertions is an important step towards a way of describing trust.

Promise theory is useful in this regard because all agents are automatically responsible for their own behaviour and only their own behaviour. Responsibility is not automatically transitive between autonomous agents: it has to be arranged through explicit agreement between agents in a controlled way; hence one avoids problems such as hidden responsibility that make the question of whether to trust an individual agent complex.

In this paper, we argue that the concept of trust can be defined straightforwardly as a *valuation* of a promise – specifically the *expectation* of autonomous behaviour. When we say that we trust something, we are directing this towards

the instigator of some promise, whether implicit or explicit. Moreover *reputation* is simply what happens to trust as it is communicated about a network, i.e. it is a 'rumour' that spreads epidemically throughout a network along different paths, and hence develops into a path-dependent estimate of trustworthiness.

The matter of evidence-gathering, in order to justify the expectation value of keeping a promise is subtle, and so we shall discuss this in some detail. We argue that there is insufficient information in the notions of trust or reputation to make a reliable estimate of trustworthiness. Thus trust is an inherently ambiguous concept; each valuation of trustworthiness is, in essence, an essentially *ad hoc* policy.

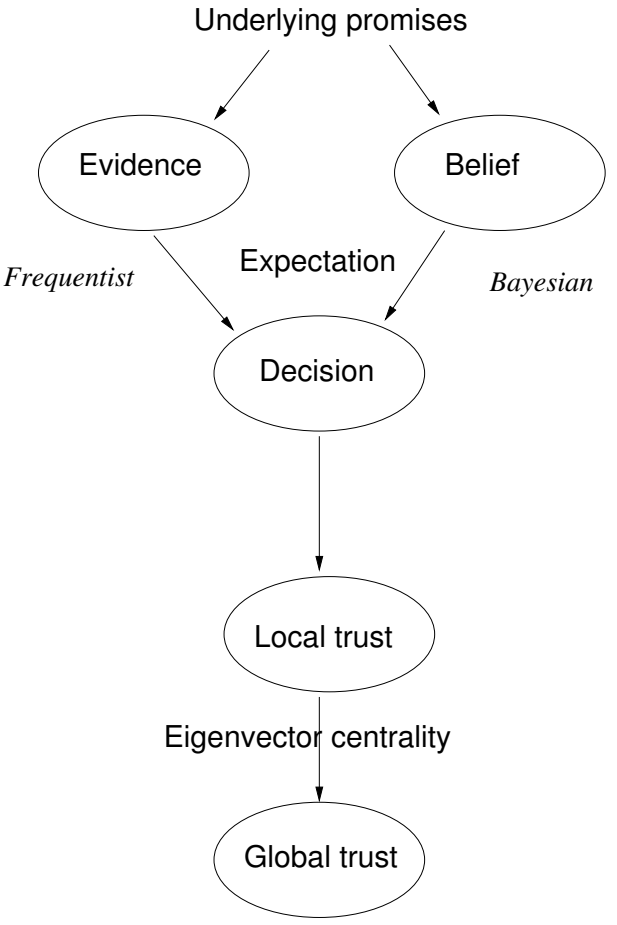


Fig. 9.1. The chain of trust from verifiable promises to local trust by an agent, to global or community trust which we interpret as reputation.

9.3 A symbiosis with promises

An agent that is known to keep its promises is considered trustworthy by any normal definition of trust i.e. the agent would be reliable and predictable such that one could put aside one's doubts about whether it might fail to live up to its assertions.

It seems natural then to associate trust with one agent's expectation of the performance of another agent in implementing its promises. This could seem like an unnecessarily narrow definition, but it turns out to be more general than one might expect. What about trust in matters that have not yet occurred? Clearly, trust could be formulated about a future *potential promise*. i.e. a promise does not have been made for us to evaluate its likely reliability. The usefulness of promises is that they encapsulate the relevant information to categorise intentions and actions.

Proposal 1 (Trust) *Trust can be defined as an* agent's expectation *that a promise will be kept. It may be assigned a value lying between 0 and 1.*

We shall define "an agent's expectation" in detail below, and we shall additionally give meaning to the concepts of when an agent is deemed to be *trustworthy* or *trusting* which are global concepts, different from merely *trusted*. This proposal has a number of positive qualities. To begin with it separates the *experiential* aspect of trust from the *nature of the actions* on which it is based. Thus in terms of philosophy of science, it makes a clean distinction between empirical knowledge (expectation) and theoretical knowledge (a promise).

Our definition is specific. The concept of trust, as normally applied in computer science is rather universal and non-specific: either one trusts another agent or one does not; however, it is seldom that we trust or distrust anyone or anything so completely. Our definition is a *typed* definition, i.e. we gauge trust separately for each individual kind of promise – and this is where promises provide a convenient notation and conceptual stepping stone. We assume that promises are a more fundamental notion than trust.

According to our definition, trust is a reliability rating made by some agent that is able to observe agents involved in a promise. We hesitate to call this a reliability *measure*: for reasons that we shall make clear, there is normally insufficient evidence on which to base a proper reliability estimate, in the sense of reliability theory[HR94].

A reputation is little more than a rumour that spreads epidemically throughout a network. Common ideas about reputation include.

- "A general opinion of someone."

- "A measure of someone's standing in the community."

Reputation is not necessarily related to trustworthiness. One could have a reputation based on how much money an agent spends, or how much fuel it uses. What characterizes a reputation, as opposed to a personal observation or evaluation, is that it is passed on. One does not observe the characteristic first hand.

Proposal 2 (Reputation) *Reputation can be defined as a valuation of some agent's past or expected behaviour that is communicated to another agent.*

We clarify and develop these basic proposals in the remainder of the paper. In particular trust will be revisited in more detail in section 8.

9.4 The literature on trust

There is an extensive literature on trust in computer science[LaP94, McI89, Win96, PJ04, JKD05, HJS04]. Much of it is concerned with generating protocols for the purpose of determining the validity of public keys and other identity tokens, or criticizing these mechanistic views in a wider security perspective. Here we are mainly concerned with general ideas about trust and reputation.

We find the recent work of Klüwer and Waaler to be of interest from the viewpoint of logic[KW05, KW06]. These authors present a natural reasoning system about trust which includes the notion of *ordering* by levels of trustworthiness.

The work that seems closest to ours may be found in ref. [BBK94] and ref. [JP05]. Here the authors distinguish between trust and reputation and provide an epidemic-like procedure for valuating the trust based on some inference rules and numerical measures that are essentially reliabilities. The calculation is hence mainly appropriate for a frequentist interpretation of probability. The authors in ref. [BBK94] are unable to distinguish trust about different issues, or relate these in their model. In ref. [JP05], an attempt is made at motivating trust types but the underlying properties of these types is not completely clear.

In our proposal:

1. We allow for multiple sources (types) for which trust and reputation are valuated.
2. Our combinatorics are based on logic and on Bayesian probability estimates, which are more appropriate estimators for the small amounts of experience involved.

Other work which we find valuable includes social viewpoints of trust (see ref. [FB01] for a review). This work brings in the matter of human value judgements, which we feel is an important issue in any definition of trust, since it

is humans who make the final decisions in practice. From a sociological viewpoint, there are many forms of currency on which to build trust. Some of these are based on the outcomes of stand-offs such as economic games, bargaining situations and so on[Axe84]. Promises have already been shown to incorporate these considerations neatly within their framework[BFb].

9.5 Common usage of trust and reputation

As with most words, the English word 'trust' has a number of related meanings which are worth documenting for reference and comparison.

- Trust implies a confidence or faith character: e.g. one "trusts in friends and family".
- It might be based on an assessment of reliability: e.g. "A trustworthy employee"
- A related, but not identical meaning has to do with presumed safety. It also means to permit something without fear. "I trust the user to access the system without stealing." Such trust can be betrayed.

  This is different because the feeling of safety is not a rationally determined quantity, whereas reliability is observable and measurable. Thus there is both a rational and an irrational aspect to trust.
- A final meaning of trust is the expression of hope, i.e. and expectation or wish: "I trust you will behave better from now on";

  Trust is therefore about the suspension of disbelief. It involves a feeling of benevolence, or competence on the part of the trustee.

  Trust of this kind expresses an acceptance of risk, e.g. a jewelry store trusts that passers-by will not smash a plate glass window very often to steal displayed goods, but rather trusts that the windows will improve sales. There could therefore be an economic decision involved in risk-taking.

Reputation is a related notion to trust. We understand this to mean a received judgement, i.e. an evaluation of an agent's reliability based on hearsay. Reputation spreads like an epidemic process, but it is potentially modified on each transmission. Thus, from a given source, several reputations might emerge by following different pathways (histories) through a network.

9.6 A general formalism for trust

Trust is somehow complementary to the idea of a service promise. This is suggested by the intuition that a promise to *use* a service implies a measure of trust on the part of the receiver. We consider trust a directed relationship from a *truster* to a *trustee*. Moreover, it is a judgement or *valuation* of a promise performed entirely by the *truster*.

We need a notation to represent this, similar to that for promises. In the spirit of the promise notation, we write the general case as:

$$S[T] \;\overset{b}{\mathbf{Trust}}\; R[U] \tag{9.1}$$

meaning that S trusts R to ensure that T keeps a promise of b to U.

In most cases, this is too much generality. In a world of autonomous agents, no agent would expect agent S to be able to ensure anything about agent T's behaviour. The more common case is therefore with only three parties

$$A_1[A_2] \;\overset{b}{\mathbf{Trust}}\; A_2[A_3] \tag{9.2}$$

i.e. agent A_1 trusts agent A_2 to keep its promise towards some third-party agent A_3. Indeed, in most cases A_3 might also be identified with A_1:

$$A_1[A_2] \;\overset{b}{\mathbf{Trust}}\; A_2[A_1] \tag{9.3}$$

which, in turn, can be simplified to

$$A_1 \;\overset{b}{\mathbf{Trust}}\; A_2. \tag{9.4}$$

In this case, trust is seen to be a dual concept to that of a promise. If we use the notation of ref. [BFb], then we can write trust as one possible valuation $v : \pi \to [0, 1]$ by A_1 of the promise made by A_2 to it:

$$A_1[A_2] \;\overset{b}{\mathbf{Trust}}\; A_2[A_1] \;\leftrightarrow\; v_1(A_2 \xrightarrow{b} A_1) \tag{9.5}$$

This is then a valuation on a par with economic valuations of how much a promise is worth to an agent[BFb]. The recipient of a promise can only make such a valuation if it knows that the promise has been made.

Proposal 3 *Trust of an agent S by another agent R can exist if agent R is informed that agent S has made a promise to it in the past, or if the recipient of the promise R is able to infer by indirect means that S has made such a promise.*

Thus any agent can formulate a trust policy towards any other agent. The only remaining question is, on what basis should such a judgement be made?

Our contention is that the most natural valuation to attach to trust is an agent's estimate of the expectation value that the promise will be kept, i.e. an estimate of the reliability of the agent's promise.

$$A_1[A_2] \ \overset{b}{\mathbf{Trust}} \ A_2[A_1] \ \overset{P}{\equiv} \ E_1(A_2 \xrightarrow{b} A_1) \tag{9.6}$$

where $\overset{P}{\equiv}$ means 'is defined by policy as', and the expectation value $E_R(\cdot)$, for agent R has yet to be defined (see Appendix A for these details). We note the essential difficulty: that such valuations of reliability are not unique. They are, in fact, entirely subjective and cannot be evaluated without ad hoc choices of a number of free parameters. We return to this point below.

9.7 Cases: The underlying promises for trust idioms

To ensure that our definition of trust is both intuitive and general, we present a number of 'use-cases' below and use these to reveal, in each case, the expectation of a promise that underlies the trust. In each case, we write the declarations of trust, in notation, in words, and as an expectation value of an underlying promise. In some cases, the expressions of trust are ambiguous and support several interpretations which can only be resolved by going to a deeper explanation in terms of promises.

- *I trust my computer to give me the right answer.* This could literally mean that one trusts the computer, as a potentially unreliable piece of hardware:

$$\mathrm{Me} \ \overset{\text{right answer}}{\mathbf{Trust}} \ \mathrm{Computer} \overset{P}{\equiv} E_{\mathrm{Me}}(\mathrm{Computer} \xrightarrow{\text{answer}} \mathrm{Me}) \tag{9.7}$$

  i.e. I expect that the computer will keep its (implicit) promise to furnish me with the correct answer.

  However, there is another interpretation. We might actually (even subconsciously) mean that we trust the company that produces the software (the vendor) to make the computer deliver the right answer when asked, i.e. I expect the promise by the vendor to me, to make the computer give me the right answer, will be kept.

$$\begin{aligned}[\mathrm{Me}][\mathrm{Computer}] \ &\overset{\text{answer}}{\mathbf{Trust}} \ [\mathrm{Vendor}][\mathrm{Me}] \\ &\overset{P}{\equiv} E_{\mathrm{Me}} \Big([\mathrm{Vendor}][\mathrm{Computer}] \xrightarrow{\text{Answer}} [\mathrm{Me}][\mathrm{Me}] \Big)\end{aligned} \tag{9.8}$$

In either case, the relationship between the promise, the expectation and the trust is the same.

- *I trust the identity of a person (e.g. by presence, public key or signature).*

  This is one of the classic problems of security systems, and we find that the simple statement hides a muddle of possibilities. It has many possible interpretations; however, in each case we obtain clarity by expressing these in terms of promises.

$$\text{Me} \overset{\text{Authentic}}{\mathbf{Trust}} \text{Signature} \overset{P}{\equiv} E_{\text{Me}}(\text{Signature} \xrightarrow{\text{Authentic}} \text{Me}) \quad (9.9)$$

In this version, we place trust in the implicit promise that a credential makes of being an authentic mark of identity. This is a simple statement, but we can be sceptical of the ability of a signature to make any kind of promise.

$$\text{Me}[\text{Signature}] \overset{\text{Authentic}}{\mathbf{Trust}} \text{Certifier}[\text{Me}] \overset{P}{\equiv} E_{\text{Me}}(\text{Certifier}[\text{Signature}] \xrightarrow{\text{Authentic}} \text{Me}) \quad (9.10)$$

i.e. I trust a Certifying Agency to ensure that the implicit promise made by the credential to represent someone is kept. Or I expect the certifying agency (possibly the originator of the signature himself) to keep a promise to me to ensure that the signature's promise to me is kept (e.g. the technology is tamper-proof).

Yet a third interpretation is that the trust of the key is based on the promise to verify its authenticity, on demand. This is the common understanding of the "trusted third party".

$$\text{Me} \overset{\text{verify key}}{\mathbf{Trust}} \text{Certifier} \overset{P}{\equiv} E_{\text{Me}} \left(\text{Certifier} \xrightarrow{\text{verify key}} \text{Me} \right) \quad (9.11)$$

i.e. I trust that the key has been authorized and is verifiable by the named Certification Agency. This last case avoids the problem of why one should trust the Certifying Agency, since it refers only to the verification service itself.

- A similar problem is encountered with currency denominations, e.g. pound notes, dollars, or Euros. These tokens are clearly not valuable in and of themselves; rather they represent value. Indeed, on British Pound notes,

the words "I promise to pay the bearer on demand the sum of ... X pounds" is still found, with the printed signature of the Chief Cashier. Indeed, the treasury would at one time, if pressed, redeem the value of these paper notes in gold. Thus trust in a ten pound note may be expressed in a number of ways.

We trust the note to be legal tender: i.e.

$$\text{Me} \overset{\text{legal}}{\mathbf{Trust}} \text{Note} \overset{P}{\equiv} E_{\text{Me}} \left(\text{Cashier} \xrightarrow{\text{gold} \wedge \text{note}} \text{Me} \right) \tag{9.12}$$

we expect that the chief cashier will remunerate us in gold on presenting the note. Alternatively, we assume that others will promise to accept the note as money in the United Kingdom (UK):

$$\text{Me} \overset{\text{legal}}{\mathbf{Trust}} \text{Note} \overset{P}{\equiv} E_{\text{Me}} \left(\text{S} \xrightarrow{\text{U(note)}} \text{Me} \right), \ \forall S \in \text{UK} \tag{9.13}$$

Interestingly neither dollars nor Euros make any much promise. Rather, the dollar bill merely claims "In God we trust"[32].

- *Trust in family and friends.*

  This case is interesting, since it is so unspecific that it could be assigned almost any meaning. Indeed, each agent is free to define its meaning autonomously. For some bundle of one or more promises $\mathcal{P}^*$ (see notation $\Rightarrow$ in section 4.1),

$$\text{Me} \overset{\mathcal{P}^*}{\mathbf{Trust}} \{\text{Family}\} \overset{P}{\equiv} E_{\text{Me}} \left(\{\text{Family}\} \overset{\mathcal{P}^*}{\Longrightarrow} A_i \right) \tag{9.14}$$

  i.e. for some arbitrary set of promises, we form an expectation about the likelihood that family and friends would keep their respective promises to the respective promisees. These promises might, in fact, be hypothetical and the evaluations mere beliefs. On the other hand, we might possess actual knowledge of these transactions, and base judgement on the word of one of these family/friend members to keep their promises to the third parties:

$$\text{Me} \overset{\mathcal{P}^*}{\mathbf{Trust}} \{\text{Family}\} \overset{P}{\equiv} E_{\text{Me}} \left(\{\text{Family}\} \overset{\mathcal{P}^*}{\Longrightarrow} \text{Me}[A_i] \right) \tag{9.15}$$

- *A trustworthy employee.*

  In this case, one bases trustworthiness is based more on a history of delivering on promises made in the context of work, e.g.:

$$\text{Boss} \overset{\text{Deliver}}{\mathbf{Trust}} \text{Employee} \overset{P}{\equiv} E_{\text{Boss}}(\text{Employee} \xrightarrow{\text{Deliver}} \text{Boss}) \tag{9.16}$$

- *I trust the user to access the system without stealing.*

  Here the promise is not to steal. The promise does not have to have been made explicitly. Indeed, in civil society this is codified into law, and hence all agents implicitly promise this by participating in that society.

- *"I trust you will behave better from now on!"*

  This can be understood in two ways. In the first interpretation, this is not so much an evaluation of trust as it is a challenge (or even warning) to the agent to do better. Alternatively, it can be taken literally as an expression of belief that the agent really will do better. In the latter case, it is:

$$\mathrm{Me} \overset{\text{Do better}}{\mathbf{Trust}} \mathrm{You} \overset{P}{\equiv} E_{\mathrm{Me}}\left(\mathrm{You} \xrightarrow{\text{Do better}} \mathrm{Me}\right) \tag{9.17}$$

9.8 Expectations of ensembles and compositions of promises

We are not done with policy's intrusion into the definition of expectation. Since promises can be composed according to straightforward rules, we must be able to compute two distinct things:

1. The expectation of a composition of promises that coexist.

2. The composition of expectations from different ensembles.

The difference between these is analogous to the difference between the combinations of experimental data into ensembles for computing probabilities, and the composition of different probable inputs in fault trees (with **AND**, **OR**, **XOR**, etc).

We have already discussed the composition of data sets into ensembles, the effect this has on probabilities, and how this is expressed in terms of the basic expectation values in section B.2

We shall have need to define the meaning of the following in order to determine the trust deriving from compound promises:

1. The expectation of incompatible promises.

2. The expectation of a composition of parallel promises between a pair of agents.

3. The expectation of a composition of serial promises between a chain of agents.

9.8.1 Parallel promise (bundle) expectation

When promises are made in parallel, the question arises as to how much to trust them as a bundle. Should one ever base one's trust on a complete package or bundle of promises? This is a subjective judgement based on whether certain promises are related in the view of the promisee. If one makes an expectation valuation for each promise individually, does it make sense to combine them as probabilities, e.g. in the manner of a fault tree[Bur04a, HR94]. One is used to the probability composition rules for binary logic of independent events.

- ($\mathbf{AND}$, $\vee$): If the promisee is dependent on several mutually reinforcing promises, then $\mathbf{AND}$ semantics are a reasonable assumption. In a security situation, this might be reasonable. The multiplicative combination rule means that each additional promise that must be in place reduces the total trust that the promiser will keep all of its promises proportionally.

- ($\mathbf{OR}$, $\wedge$) Here one says that if one or more promises are kept, then trustworthiness is reinforced. This is an optimistic policy which seems to suggest that the promisee is understanding about the promiser's potential difficulties in keeping a promise.

- ($\mathbf{XOR}$): An alternative scenario is to have a number of promises that are alternatives for one another. For instance, mutually exclusive conditional promises that behave like a switch: e.g.

$$S \xrightarrow{x \ \mathbf{XOR} \ x'} R \equiv \begin{cases} S \xrightarrow{x|y} R \\ S \xrightarrow{x'|\neg y} R \end{cases} , \tag{9.18}$$

  i.e. S promises x to R, iff y, else it promises x'.

- (RANKED) If the promises are ranked in their importance to the recipient, then the measure of trust associated with the package is best judged by weighting the importance appropriately. Referring to the discussion in section B.2, this admits a general convex combination of contributions for ranking an $\mathbf{OR}$ (see below).

Let us consider how these are represented as functions.

Definition 59 (Expectation of a promise bundle) *Let S (sender) and R (recipient) be agents that make a number of promises in parallel, the composition of a bundle of parallel promises $S \xrightarrow{b^*} R$ is a function F_R of the expectations of the individual promises:*

$$E_R\left(S \xrightarrow{b^*} R\right) \stackrel{P}{\equiv} F_R\left(E_R\left(S \xrightarrow{b_1} R\right), E_R\left(S \xrightarrow{b_2} R\right), \ldots\right) \tag{9.19}$$

The function F_R is a mapping from N promise expectations to a new expectation value:

$$F_R : [0,1]^N \to [0,1] \tag{9.20}$$

Several such functions are known from reliability theory, e.g. in fault tree analysis (see for instance ref. [HR94]). Examples include,

$$F_R^{\text{AND}}\left(S \xrightarrow{b^*} R\right) = \prod_i E_R\left(S \xrightarrow{b_i} R\right) \tag{9.21}$$

$$\begin{aligned} F_R^{\text{OR}}\left(S \xrightarrow{b^*} R\right) &= 1 - \prod_i \left(1 - E_R\left(S \xrightarrow{b_i} R\right)\right) \\ &\simeq \sum_i E_R\left(S \xrightarrow{b_i} R\right) \pm O(E^2) \end{aligned} \tag{9.22}$$

$$\begin{aligned} F_R^{\text{XOR}}\left(S \xrightarrow{b^*} R\right) &\simeq 1 - \prod_i \left(1 - E_R\left(S \xrightarrow{b_i} R\right)\right) \\ &\simeq \sum_i E_R\left(S \xrightarrow{b_i} R\right) \pm O(E^2). \end{aligned} \tag{9.23}$$

where $O(E^2)$ denotes terms or the order of the probability squared, which are small. A further possibility is to take a weighted mean of the promise estimates. This better supports the view in section B.2 about different sizes ensembles and their relative weights. There might be additional (irrational) reasons for giving priority to certain promises, e.g. leniency with respect to a difficult promise.

To combine the different possibilities (analogously to fault trees) one could first reduce products of **AND** promises into sub-bundles, then recombine these using a weighted estimate.

$$\begin{aligned} F_R^{RANKED} &\stackrel{P}{\equiv} \sum_i \alpha_i E_R\left(S \xrightarrow{b_i} R\right) \\ \sum_i \alpha_i &= 1 \end{aligned} \tag{9.24}$$

Note that, due to the reasoning of probability theory, the expectation of something AND something else is less than the probability of either. This might be seen as pessimistic as far as trust is concerned. We have to make a policy decision about whether or not to place any weight on the combined expectation of a bundle of promises, or whether to decide to only allow individual expectations.

For example, suppose an agent makes two contradictory promises about services levels, e.g. promise to respond in 4ms and promise to respond in 5ms.

$$\begin{aligned} S &\xrightarrow{4} R \\ S &\xrightarrow{5} R \end{aligned} \tag{9.25}$$

Formally, this is a conflict, since both promises cannot be true at the same time. The trust in each individual promise can be estimated independently for the two promises. The agent reliability expectations of delivering "4" or "5" units of service are:

$$R \overset{4}{\mathbf{Trust}} S = E_R(4) = p(4) = 0.1 \tag{9.26}$$

$$R \overset{5}{\mathbf{Trust}} S = E_R(5) = p(5) = 0.2 \tag{9.27}$$

Then we can consider what the expectation of the combination of promises is. If the agent S makes both promises simultaneously, the expectation of the combined promises will be:

$$E_R(4 \,\mathbf{XOR}\, 5) \simeq \frac{(e_4\, E_R(4) + e_5\, E_R(5))}{(e_4 + e_5)} \tag{9.28}$$

where e_4 is our estimate of likelihood the agent can deliver "4" and e_5 is the estimate of likelihood of delivering "5". These beliefs can be based on many potential sources of information, chosen as a matter of policy; one possibility is to simply identify $e_4 \overset{P}{\equiv} E_R(4)$ and $e_5 \overset{P}{\equiv} E_R(5)$. Thus a simple policy solution could be to take

$$E_R(4 \,\mathbf{OR}\, 5) \overset{P}{\equiv} \frac{E_R(4)^2 + E_R(5)^5}{E_R(4) + E_R(5)} = 0.17 \tag{9.29}$$

i.e. in general a sum of squares.

9.8.2 Incompatible promise expectation

For incompatible promises we must have at least complementary behaviour (NOT):

$$\begin{aligned} E_A(S \xrightarrow{\neg b} R) &= 1 - E_A(S \xrightarrow{b} R) \\ F_R(E_R(S \xrightarrow{\neg b} R)) &= 1 - F_R(E_R(S \xrightarrow{b} R)) \end{aligned} \tag{9.30}$$

Ideally incompatible promises would not be made, without conditionals to select only one of the alternatives.

In the case of **AND** it is necessary already to resolve the ambiguity in the meaning of the combination of incompatible promises. It is by definition a logical impossibility for incompatible promises to be kept. Thus, while we cannot prevent an agent from promising such nonsense, our expectation of the combination ought to be zero.

Definition 60 (Expectation of incompatible promises with AND) *The expectation of incompatible promises,*

$$F_R\left(A_1 \xrightarrow{b_1} A_2 \ \mathbf{AND}\ A_1 \xrightarrow{b_2} A_2\right) \equiv 0 \ \text{ when } b_1 \# b_2 \tag{9.31}$$

is defined to be zero for any rational agent.

Hence, in the example above,

$$E_R(4\ \mathbf{AND}\ 5) = 0. \tag{9.32}$$

9.8.3 Serial promise expectation and transitivity of trust

Several systems base their operation on the idea that trust is to some extent transitive. "The Web of Trust" notion in public key management idea proposes that trust can be conferred transitively. This is not a property of promises, so it is of interest to consider how this works. In other words, if A_1 trusts A_2 to do b, and A_2 trusts A_3 to do b, then A_1 will often trust A_3 to do b. Here b is generally taken to be "reveal one's true identity". This notion does not fit well with a promise theory interpretation of trust because it is type-unspecific.

This is easy to see by noting that

$$A_1 \xrightarrow{b} A_2, A_2 \xrightarrow{b} A_3 \not\Rightarrow A_1 \xrightarrow{b} A_3 \tag{9.33}$$

i.e. if A_1 makes a promise of b to A_2 and A_2 makes the same promise to A_3, it does not follow that A_1 has made any promise to A_3.

An unspecific trust model might conform to the following property:

$$(i)\ \ (A_1 \text{ Trusts } A_2), (A_2 \text{ Trusts } A_3) \Rightarrow A_1 \text{ Trusts } A_3 \tag{9.34}$$

In terms of promises, we would interpret this to mean that, if A_1 trusts A_2 (to keep promises to A_1) and A_2 trusts A_3 (to keep promises to A_2) then A_1 should trust A_3 to keep promises to A_1. This is far from being a rational policy, since

there is no evidence passed on about the reliability of agents. A less problematic alternative is:

$$(ii)\ \ (A_1 \overset{\text{inform}}{\mathbf{Trust}} A_2), (A_2 \overset{b}{\mathbf{Trust}} A_3) \Rightarrow A_1[A_3] \overset{b}{\mathbf{Trust}} A_3[A_2] \quad (9.35)$$

If A_1 trusts A_2 (to inform it about its relations with A_3) and A_2 trusts A_3 (to keep its promise of b to A_2), then A_1 trusts that A_3 is trustworthy in its promise of b to A_2.

The matter of serial promises is one of diverging complication. We make some brief notes about the problems associated with serial promises, and leave the potentially extensive details for elsewhere. The problems with trusting a distributed collection of promises are

1. Promises are not common knowledge, so we do not have all the information.
2. Promises are not transitive.

Knowledge about the promises and the local evaluations by the agents can only be guaranteed by making chains of promises between the agents to share this knowledge.

$$\begin{array}{ccccc} A_1 & \xrightarrow{\text{tell rep}} & A_2 & \xrightarrow{\text{tell rep}} & A_3 \\ A_1 & \overset{\pi:U(\text{tell rep})}{\longleftarrow} & A_2 & \overset{\pi:U(\text{tell rep})}{\longleftarrow} & A_3 \end{array} \quad (9.36)$$

In order to pass on the necessary information about trust to a third party, it must be relayed. Expectation of a chain of promises depends on a chain of such trust and Use(trust) promises. However, each agent in the chain agrees only to trust the previous agent. There is no automatic agreement to trust the previous members. If one were to make an explicit promise to trust each agent's information about trust, this would require a promise graph like the one in fig. 9.2. In order to remove the ambiguity of the trust promises, we must use a different *promise type* for trust about each agent in the graph. i.e. the trust passed on from agent a must retain this label in being transferred. However, here one has a paradox: if an agent is potentially unreliable, then it can easily lie about this information. Such serial chains are, in general fraught with uncertainty, thus agents might well choose, as a matter of policy, to disregard reputations.

9.9 How promises guide expectation

What kind of policy should be employed in defining the expectation of future behaviour? Probability theory is built on the assumption that past evidence can

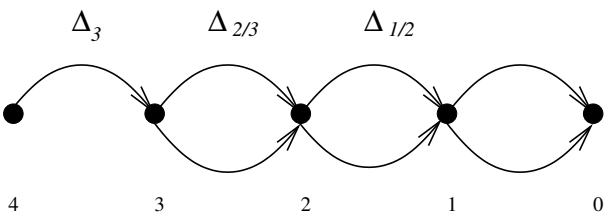


Fig. 9.2. A chain of trust promises to transfer some valuation of trust in one direction (only), from node a to each agent up to node d. This method is unreliable because nodes b and c are under no obligation to pass on the correct value. Note that these are promise arrows, not trust arrows.

This is clearly a fragile and somewhat complicated structure. An alternative approach is to avoid chains of greater length than one, and also eliminate the extraneous and essentially impotent promises from the chain, as in fig. 9.3. However, this leads us merely back to the notion of a centralization, either in the form of a trusted party for all agents, or as a complete peer-to-peer graph.

motivate a prediction of the future. At the heart of this is an assumption that the world is basically constant. However, future prediction is the essence of gambling: there are scenarios in which evidence of the past is not an adequate guide to future behaviour. An agent might also look elsewhere for guidance.

- *Initialization*: An agent of which we have initially no experience might be assigned an initial trust value of 1, $\frac{1}{2}$, or 0 if we are respectively trusting, neutral or un-trusting by nature.
- *Experience*: One's own direct experience of a service or promise has primacy as a basis for trusting an agent in a network. However, an optimistic agent might choose not to allow the past to rule the future, believing that agents can change their behaviour, e.g. "the agent was having a bad day".
- *Advice*: An agent might feel that it is not the best judge and seek the advice of a reputable or trustworthy agent. "Let's see what X thinks". We shall use this idea in section 9.11 to define a global trustworthiness.
- *Reputation*: Someone else's experience with a promise can serve as an initial value for our own trust.
- *Damnation*: Some agents believe that, if an agent fails even once to fulfil a promise, then it is completely un-trustworthy. This extreme policy seems excessive, since there might be reasons beyond the control of the agent that prevent it from delivering on its promise.

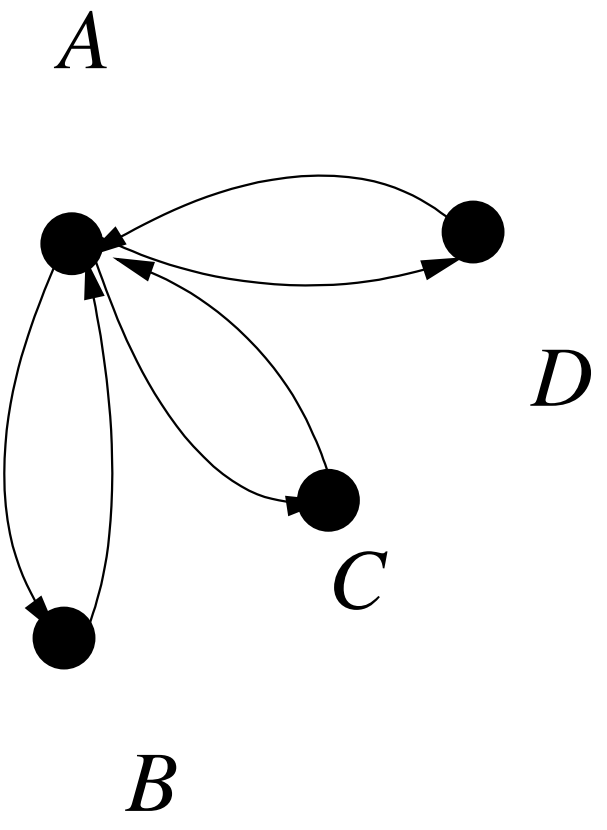


Fig. 9.3. A more reliable approach of passing on the trust node a holds on to nodes b, c and d.

If we lack any evidence at all about the trustworthiness of an agent with respect to a given promise, we might adopt a policy of using the agent's record of keeping other kinds of promises.

Proposal 4 (Transference of evidence) *In the absence of direct evidence of type $t(b)$, in a promise body b, one may use a policy determined mixture of values from other types as an initial estimate.*

The rationality of such a procedure can easily be questioned, but there is no way to rule out the ad hoc decision as a matter of policy.

9.10 Reputation

We have defined a reputation to be simply a valuation of something (not necessarily a promise) received by an agent about some other agent. A natural basis for reputation (and one that is used on 'reputation systems' in computing) is the valuation of trustworthiness. Here we consider the effect that such transmission of information has on the local trust within a network of agents.

9.10.1 Borrowed trust

Suppose that and agent T trusts an agent S to keep its promise to R with probability $E_T\left(S \xrightarrow{b} R\right)$, and suppose that this agent T promises to transmit this as S's reputation to another agent U. U's estimate of the trustworthiness of T's communication is

$$U \overset{\text{reputation}}{\mathbf{Trust}} T \overset{P}{\equiv} E_U\left(T \xrightarrow{\text{reputation}} U\right) \tag{9.37}$$

Can we say what U's expectation for the reliability of the original promise $a \xrightarrow{b} c$ should be? In spite of the fact that probabilities for independent events combine by multiplication, it would be presumptuous to claim that

$$E_U\left(S \xrightarrow{b} R\right) = E_U\left(T \xrightarrow{\text{reputation}} U\right) E_T\left(S \xrightarrow{b} R\right), \tag{9.38}$$

since U does not have any direct knowledge of $E_T\left(S \xrightarrow{b} R\right)$, he must evaluate the trustworthiness and reliability of the source.

Suppose we denote the communicated value of $E_T\left(S \xrightarrow{b} R\right)$ by $\mathcal{E}_{U \leftarrow T}\left(S \xrightarrow{b} R\right)$, then one could conceivably (and as a matter of rational policy) choose to define

$$E_U\left(S \xrightarrow{b} R\right) \overset{P}{\equiv} E_U\left(T \xrightarrow{\text{reputation}} U\right) \mathcal{E}_{U \leftarrow T}\left(S \xrightarrow{b} R\right). \tag{9.39}$$

With this notation, we can conceivably follow historical paths through a network of promises.

However, it is important to see that no agent is obliged to make such a policy. Thus trust and reputation do not propagate in a faithfully recursive manner. There is, moreover, in the absence of complete and accurate common knowledge by all agents, an impossibility of eliminating the unknowns in defining the expectation values.

9.10.2 Promised trust

Trust is an evaluation that is private to an agent. This evaluation can be passed on in the form of a communication (leading to reputation), or it can be passed on as a promise to trust.

- S promises R that S will trust R: $S \xrightarrow{\tau=0.6} R$.
- S promises R that S will trust T: $S \xrightarrow{\tau=0.6} R[T]$.

Why would anyone promise a party (R) to trust T without telling R? One reason is that there might be strategic bargaining advantages to doing this[sch60].

9.10.3 Updating trust with reputation

An agent can use the reputation of another agent as a sample of evidence by which to judge its trustworthiness. It can then attach a certain weight to this information according to its belief, in order to update its own trust. The weighted addition modifies the old trust value T with the new reputation data R.

$$E \mapsto \frac{w_{\rm new} R + w_{\rm old} T}{w_{\rm new} + w_{\rm old}} \tag{9.40}$$

This is indistinguishable from a Bayesian update.

9.11 Global Measures of Trust

Which are the most trusted agents in a network? Trust has so far been measured at the location of each individual agent. The valuation is private. A trust valuation becomes an agent's reputation when the valuation is passed on to others. The passing-on includes a revisional belief process too; this is also a Bayesian posterior probability update process, just like the case of basing trust on different ensembles in section B.2.

Let us postulate the existence of a vector of received trusts that is available to any particular agent. The agent is then able to combine this information to work out a global measure, which we can call *community trust*. This is analogous to the graphical security model in [BCE04a].

The trust matrix T is defined as follows. The (A, B)-th element of the matrix

$$T_{AB}(b) \equiv E_A(B \xrightarrow{b} *) \tag{9.41}$$

is A's trust in B with respect to all promises of type b.

Definition 61 (Community trust (Trustworthiness and trustingness)) *The global or community trust is defined by the principal eigenvectors of T and T^{T}. Since this is a transmitted quantity by definition it is a reputation.*

The global reputations for being trustworthy $\vec{W}$ *are defined by the normalized components of the principal eigenvector of the transpose matrix:*

$$T_{BA}W_B = \lambda W_A. \tag{9.42}$$

The global reputations for being most trusting $\vec{S}$ *are defined by the normalized components of the principal eigenvector*

$$T_{AB}S_B = \lambda S_A. \tag{9.43}$$

An agent is said to be trusting if it assigns a high probability of keeping its promises to those agents that it trusts. An agent is said to be trustworthy if other agents assign it a high probability of keeping promises to it.

Observe that, in the absence of labels about specific agent relationships, the concepts of *trustworthiness* and *trustingness* for an agent A are properties of the global trust graph that has A as a source, and not of an individual agent, since they are derived from relationships and by voting.

We can easily show that this has the property of a proportional vote. Let v_i denote a vector for the trust ranking, or connectedness of the trust graph, of each node i. Then, the trustworthiness of node i is proportional to the sum of the votes from all of i's nearest neighbours, weighted according to their trustworthiness (i.e. it is just the sum of their trust valuations):

$$v_i \propto \sum_{j=\text{neighbours of } i} v_j \ . \tag{9.44}$$

This may be more compactly written as

$$v_i = (\text{const}) \times \sum_j T_{ij} v_j \ , \tag{9.45}$$

where T is the *trust graph adjacency matrix*, whose entries T_{ij} are 1 if i is a neighbour of j, and 0 otherwise. We can rewrite eqn. (9.45) as

$$T\,\vec{v} = \lambda \vec{v} \ . \tag{9.46}$$

Now one sees that the vector is actually an eigenvector of the trust matrix T. If T is an $N \times N$ matrix, it has N eigenvectors (one for each node in the network), and correspondingly many eigenvalues. The eigenvalue of interest is the principal eigenvector, i.e. that with highest eigenvalue, since this is the only one

that results from summing all of the possible pathways with a positive sign. The components of the principal eigenvector rank how self-consistently 'central' a node is in the graph. Note that only ratios v_i/v_j of the components are meaningfully determined. This is because the lengths $|\vec{v}| = \sqrt{\sum_i v_i v_i}$ of the eigenvectors are not determined by the eigenvector equation. We normalize them here by setting the highest component to 1. This form of well-connectedness is termed 'eigenvector centrality' [Bon87] in the field of social network analysis, where several other definitions of centrality exist.

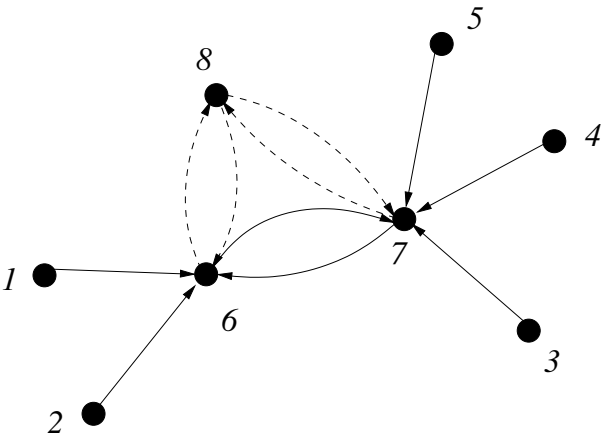


Fig. 9.4. An example trust graph. For simplicity all trust arrows are assumed of the same type, e.g. trust in the promise to pay bills. Dashed lines are lines which will be removed in the second example.

Note this does not assume any transitivity of trust, it says simply: each agent's trust worthiness is equal the sum of all the other agents' trust measures (as if they are voting), weighted so that the most trustworthy agents' opinions are weighted proportionally highest. It is a proportional representation vote by the agents about one another.

9.11.1 Example of global trust

Consider a number of promises of a single type, e.g. agents promise to pay their bills in various service interactions. Each payee then rates its expectation of the payer and makes this information globally available as a public measure of its

local trust. Referring to fig. 9.4, we assume the following local trusts:

$$\begin{aligned}
1 &\xrightarrow{\tau:\text{pay}} 6 = 0.2 \\
2 &\xrightarrow{\tau:\text{pay}} 6 = 0.3 \\
3 &\xrightarrow{\tau:\text{pay}} 7 = 0.1 \\
4 &\xrightarrow{\tau:\text{pay}} 7 = 0.1 \\
5 &\xrightarrow{\tau:\text{pay}} 7 = 0.1 \\
6 &\xrightarrow{\tau:\text{pay}} 7 = 0.6 \\
7 &\xrightarrow{\tau:\text{pay}} 6 = 0.5 \\
6 &\xrightarrow{\tau:\text{pay}} 8 = 0.8 \\
8 &\xrightarrow{\tau:\text{pay}} 6 = 0.2 \\
7 &\xrightarrow{\tau:\text{pay}} 8 = 0.8 \\
8 &\xrightarrow{\tau:\text{pay}} 7 = 0.3
\end{aligned} \tag{9.47}$$

The trust matrix is thus

$$T = \left(\begin{array}{cccccccc}
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.1 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.6 & 0.8 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.8 \\
\hline
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 & 0.3 & 0.0
\end{array}\right) \tag{9.48}$$

Note that the bars delineate the dashed lines which will be removed in the second example. The normalized right eigenvector $\vec{S}_8$ represents how trusting the agents are. The left eigenvector $\vec{W}_8$ (or the eigenvector of the transpose matrix) represents the global trustworthiness:

$$\vec{S}_8 = \begin{pmatrix} 0.21 \\ 0.31 \\ 0.10 \\ 0.10 \\ 0.10 \\ 1.00 \\ 0.94 \\ 0.50 \end{pmatrix}, \quad \vec{W}_8 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.55 \\ 0.65 \\ 1.00 \end{pmatrix} \tag{9.49}$$

Thus, agent 8 is the most trustworthy. Agents 1 to 5 are not trustworthy at all in this scenario, since we have not rated any promises made by them. Agent

6 is the most trusting of all, since it gives a large amount of trust to agent 8. Thus, these two agents colour the global picture of trust significantly through their behaviours.

We note that the agents with zero trust ratings are all recipients of promises; they do not make any promises of their own. These are suppliers of whatever service or good is being sold; they do not promise payments to anyone, hence no one needs to trust them to pay their bills. The reader might find this artificial: these agents might make it their policy to trust the agents even though they have made no promise. In this case, we must ask whether the trust would be of the same type or not: i.e. would the buyers trust the suppliers to pay their bills, or would their trust be based on a different promise, e.g. the promise to provide quality goods.

By contrast, the agents who are not trusted are somewhat trusting by virtue of receiving such promises of payment.

Suppose we eliminate agent number 8 (by removing the dashed lines in the figure), let us see how the ranking changes when we delete this important agent. Now agent 6 still remains the most trusting, but agent 7 becomes the most trusted, once again mainly due to agent 6's contribution.

$$\vec{S}_7 = \begin{pmatrix} 0.37 \\ 0.55 \\ 0.17 \\ 0.17 \\ 0.17 \\ 1.00 \\ 0.92 \end{pmatrix}, \quad \vec{W}_7 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.91 \\ 1.00 \end{pmatrix} \tag{9.50}$$

We can note that the symmetries of the graph are represented in the eigenvector in a natural way.

9.11.2 Boundaries and allegiances

Canright and Monsen have defined regions of a graph, based on the structures that arise naturally from eigenvector centrality[CEM04]. This has been further developed for directed graphs in ref. [BCE04b]. Trust is sometimes associated with maintaining certain boundaries or allegiances. The global trust model proposed above falls into a natural landscape based on the graph, that is characterized by local maxima. Agents cluster naturally into distinct hills of mutual trust, separated by valleys of more tenuous trust, in the centrality function.

This characterization is a useful way of identifying a community structure. Humans are not very good at understanding boundaries: they understand identities. e.g. a company name, but where is the real boundary of the company or computer system? Its tendrils of influence might be farther or closer than one imagines. The topology of underlying promises offers a quantifiable answer to this question. Such allegiances can be compared to the notion of a coalition in game theory[NM44, Rap70].

9.12 Trust architectures

Trust is closely associated with information dissemination. There are essentially only two distinct models for achieving information distribution: centralization and *ad hoc* epidemic flooding. Alternatively one might call them, central-server versus peer-to-peer.

Two so-called trust models are used in contemporary technologies today, reflecting these approaches: the Trusted Third Party model (e.g. X.509 certificates, TLS, or Kerberos) and the Web of Trust (as made famous by the Pretty Good Privacy (PGP) system due to Phil Zimmerman and its subsequent clones). Let us consider how these models are represented in terms of our promise model.

9.12.1 Trusted Third Parties

The centralized solution to "trust management" is the certificate authority model, introduced as part of the X.509 standard used in web authentication and modified for a variety of other systems (See fig. 9.5)[IT93, Rec97, HPFS02]. In this model, a central authority has the final word on identity confirmation and often acts as a broker between parties, verifying identities for both sides.

Definition 62 (Authority) *An agent which is the source of a promise and whose word is beyond doubt (i.e. a trusted party).*

An central authority promises (often implicitly) to all agents the legitimacy of each agent's identity (hopefully implying that it verifies this somehow). Moreover, for each consultation the authority promises that it will truthfully verify an identity credential (public key) that is presented to it. The clients and users of this service promise that they will use this confirmation. Thus, in the basic

interaction, the promises being made here are:

$$\text{Authority} \xrightarrow{\text{Legitimate}} \text{User} \tag{9.51}$$

$$\text{Authority} \xrightarrow{\text{Verification}} \text{User} \tag{9.52}$$

$$\text{User} \xrightarrow{U(\text{Verification})} \text{Authority} \tag{9.53}$$

To make sense of trust, we look for expectations of the promises being kept.

1. The users expect that the authority is legitimate, hence they trust its promise of legitimacy.
2. The users expect that the authority verifies identity correctly, hence they trust its promise of verification and therefore use it.

Users do not necessarily have to be registered themselves with the authority in order to use its services, so it is not strictly necessary for the authority to trust the user. However, in registering as a client a user also promises its correct identity, and the authority promises to use this.

$$\text{User} \xrightarrow{\text{Identity}} \text{Authority} \tag{9.54}$$

$$\text{Authority} \xrightarrow{U(\text{Identity})} \text{User} \tag{9.55}$$

One can always discuss the evidence by which users would trust the authority (or third party). Since information is simply brokered by the authority, the only right it has to legitimacy is by virtue of a reputation. Thus expectation 1. above is based, in general, on the rumours that an agent has heard.

Most of the trust is from users to the authority, thus there is a clear subordination of agents in this model. This is the nature or centralization.

9.12.2 Web of Trust

Scepticism in centralized solutions (distrust perhaps) led to the invention of the epidemic trust model, known as the Web of Trust (see fig. 9.6)[AR97]. In this model, each individual agent is responsible for its own decisions about trust. Agents confirm their belief in credentials by signing one another's credentials. Hence if I trust A and A has signed B's key then I am more likely to trust B.

As a management approximation, users are asked to make a judgement about a key from one of four categories: i) definitely trustworthy, ii) somewhat trustworthy, iii) un-trustworthy, iv) don't know.

An agent then compares these received valuations to a threshold value to decide whether or not a credential is trustworthy to it.

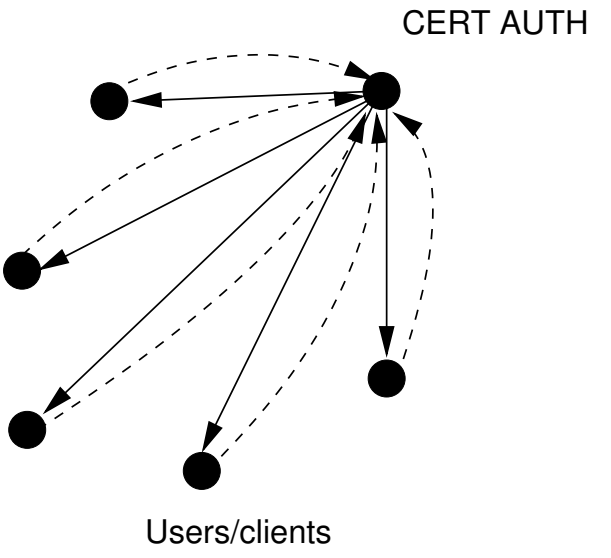


Fig. 9.5. The Trusted Third Party, e.g. TLS or Kerberos. A special agent is appointed in the network as the custodian of identity. All other agents are expected to trust this. The special agent promises to verify the authenticity of an object that is shared by the agents. In return for this service, the agents pay the special agent.

The promises are between the owner of the credential and a random agent:

$$\text{Owner} \xrightarrow{\text{Identity}} \text{Agent} \tag{9.56}$$

$$\text{Agent} \xrightarrow{U(\text{Identity})} \text{Owner} \tag{9.57}$$

$$\text{Agent} \xrightarrow{\text{Signature}} \text{Owner} \tag{9.58}$$

$$\text{Owner} \xrightarrow{U(\text{Signature})} \text{Agent} \tag{9.59}$$

The owner must first promise its identity to an agent it meets. The agent must promise to believe and use this identity credential. The agent then promises to support the credential by signing it, which implies a promise (petition) to all subsequent agents. Finally, the owner can promise to use the signature or reject it. Trust enters here in the following ways:

1. The agent expects that the identity of the owner is correct and trusts it. This leads to a Use promise.

2. The Owner expects that the promise of support is legitimate and trusts it. This leads to a Use promise.

What is interesting about this model is that it is much more symmetrical than the centralized scheme. It has certain qualities that remind us of our definition of global trust in section 9.11. However, it is not equivalent to our model, since

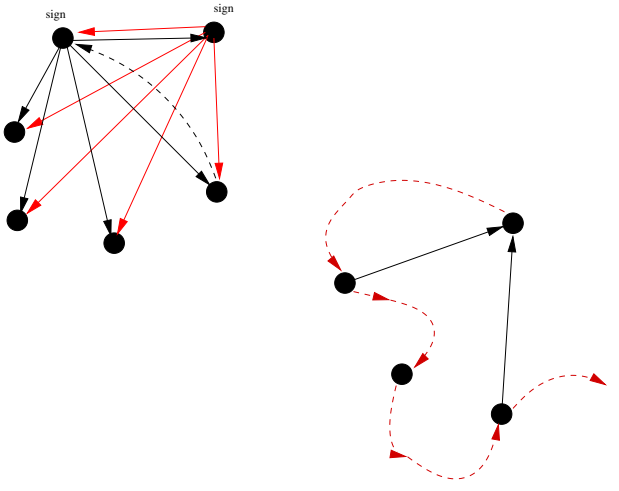


Fig. 9.6. In a web of trust an agent signals a promise to all other agents that it has trusted the authenticity of the originator's identity. As a key is passed around (second figure) agents can promise its authenticity by signing it or not.

the very nature of the web of trust is dictated by the transactions in the model, which are automatically bilateral (ours need not be). Moreover, the information is passed on in a peer to peer way, where as our global idealization makes trust valuations common knowledge (global reputations). In some respects, the web of trust is a pragmatic approximation to the idealized notion of trust in section 9.11. The main differences are:

- In the Web of trust, a limited number of expectation values is allowed and the user does not control these, i.e. there are few policy choices for agent expectation allowed.
- An agent does not see a complete trust or promise graph. It sees only the local cluster to which it is connected. This is sufficient to compute a global trust for that component of the graph.
- The Web of Trust graph is always bilateral, with arrows moving in both directions, thus no one is untrusted, or un-trusting.
- The information to construct a fully self-consistent measure of trust is not available in the system. Hence there is no clear measure of who is more

trustworthy in the web of trust.

Some of these limitations could no doubt be removed. A Bayesian approach could naturally lead to a better approximation. However, a basic flaw in these implementation mechanisms is the need to trust of the mediating software itself. Since, as we have shown, trust is not necessarily transitive, one ends up in most cases trusting the software that is supposed to implement the trust management rather than the parties themselves.

9.13 Summary

The concept of promises provides a foundation that has been unclear in discussions of trust. It allows us to decouple the probabilistic aspect from the network aspect of policy relationships, without introducing instantaneous events. It provides (we claim) a natural language for specific policies, extended over time. Promises have types and denote information flow which in turn allows us to discuss what is trusted and by whom. We believe the use of promises to be superor to a definition based on actions, since the localization of actions as space-time events makes trust ill-defined if the action has either not yet been executed or after it has been executed.

Promises allow us to relate trust and trust-reputation in a generic way, and suggest an algorithm from which to derive global network properties, based on social network theory. This is a significant improvement over previous models. Reputation is not uniquely coupled to trust, of course – it can be related to many different valuations of promised behaviour, including wealth, kindness etc.

We show how bundles of promises can be combined using the rules for probabilistic events (similar to fault tree analysis) and we model the two main trust architectures easily. The PGP Web of Trust as well as the Trusted Third Party can be explained as a special case the global trust models derived here; however standard tools do not permit users to see the entire web, or measure relative trust-worthiness in a community using these implementations.

In future work there is the possibility to use this notion of trust in explicit systems. The Unix configuration system cfengine[Bur93] uses the notion of promises and agent autonomy to implement a policy based management system. The trustworthiness of hosts with respect to certain different behaviours can be measured directly by neighbouring agents to whom promises are made. More generally, if one has a monitoring system that one believes trustworthy to begin with, it is possible to observe whether an agent stops keeping its own promises about security issues. This might be a signal to reevaluate one's expectation

that the system is trustworthy. These tests have been partially imeplemented in cfengine and are presently being tested.

Trust is merely an expression of policy and it is therefore fundamentally *ad hoc*. Promises reveal the underlying motives for trust and whether they are rationally or irrationally formed.

Notes

[32]It is a matter of belief whether one assigns this trust to a promise made by an agent called God.

Part 2
Applications

10
Policy, Autonomy, and Voluntary Cooperation

We believe that Promise Theory has the opportunity to explain wide ranging and important phenomena from information science to biology, society and economics. However, these broad topics lie beyond the reach of the present book. To reach those heights, one needs to master the basic patterns and forms of reasoning for using promises in an engineering sense.

The opportunity for any atomic theory is to explain how fundamental building blocks can be assembled into something that seems to exceed the sum of those parts. The fusion of semantics and dynamics in μ-promises easily lends itself to this agenda.

In this chapter, we hope to lay some foundations for modelling with promises that allows simple insights and predictions about systems to be understood.

10.1 The policy timescale

A policy is a declaration, summarizing a number of resolutions made by an agent. The function of a policy is to gather the results of a complex decision-making process into a single convenient summary of intent. A policy can be thought of as defining what is admissable or acceptable. It can therefore be regarded as a *constraint* on the domain of possible intentions for the agent.

If we say 'policy' rather than 'decision', it implies more than a passing determination, or spur-of-the-moment choice. A policy is expected to have a more lasting value. Perhaps it enshrines a guiding principle for the individual, which will preside over many situations and encounters. Just as rules and laws are the cached results of judicial process, policy is a way of avoiding the need to arrive at a similar decision in each new case. Thus, by assumption policy is considered to be *slowly changing*.

Using promise theory, we can develop a notion of distributed or entangled policy for multiple agents working together, using only the pure policies of individual agents. A policy is merely a superposition of possible promises.

So policy can be tailored to the different roles and circumstances in a division of labour, with agents at possibly different times and locations in a collaborative network.

This, however, leads to the possibility of contradiction: part of policy is about how agents interact, and once agents can interact, two agents can interact together in a way that is acceptable to them, but not to a third party. This problem of *policy conflict* is the pernicious problem that has serious problems in an obligation picture, but which we shall resolve simply in the promise picture.

10.2 Policy versus decision theory

Policy should not be confused with optimization or rational decision theory. In decision theory, or game theory, decisions are made rationally by maximizing a utility function[NM44]. A policy decision is distinct from an optimization because it is made by agents, who can decide to behave either rationally or irrationally, and might not have decided on an appropriate utility function.

In optimization one tries to *compute* the best course of action, using a predecided criterion or model. The result of the optimization is a rational computation, with (hopefully) only one answer—the element of decision has been removed by a standardized procedure. There are several meta-level policy decisions surrounding this however: the decisions about whether to use optimization or not, which model is correct, and which answer (if there are several) to use, are matters for policy to decide. policy can therefore be informed and guided by optimization, but in the final analysis the choice is *ad hoc*.

The construction of a distributed policy from individual agent promises is precise and can be understood graphically. Such an atomic reconstruction provides a framework in which to reason about the distributed effect of policy. Immediate applications include resolving the problem of policy conflicts in autonomous computer networks.

10.3 Rule and law, Norms and requirements

One of the problems in discussing policy based management of distributed systems[SM93, LS96] is the assumption that all agents will follow a consistent set of rules. For this to be true, we need either an external authority to impose a consistent policy

from a bird's eye view, or a number of independent agents to collaborate in a way that settles on a 'consistent' picture autonomously.

Political autonomy is the key problem that one has to deal with in ad hoc / peer-to-peer networks, and in pervasive computing. When the power to decide policy is delegated to individuals, orders cannot be issued from a governing entity: consistency and concensus must arise purely by *voluntary cooperation*. There is no current model for discussing systems in this situation.

In this chapter outline a theory for the latter, and in doing so provides a way to achieve the former. The details of this theory require a far more extensive and technical discussion than may be presented in this short contribution; details must follow elsewhere.

It has been clear to many authors that the way to secure a clear and consistent picture of policy, in complex environments, is through the use of formal methods. But what formalism do we have to express the necessary issues? Previous attempts to discuss the consistency of distributed policy have achieved varying degrees of success, but have ultimately fallen short of being useful tools except in rather limited arenas. For example:

- Modal logics: these require one to formulate hypotheses that can be checked as true/false propositions. This is not the way most planners or designers work.

  It is a stress-testing procedure or a destructive testing method.

- The π-calculus: has attractive features but focuses on issues that are too low-level for management. It describes systems in terms of states and transitions rather than policies (constraints about states)[Par01].

- Implementations like IPSec[FW01, SAB+01], Ponder[DDLS00] etc. these do not take explicitly into account the autonomy of agents and thus while these implement policies well enough, they are difficult to submit to analysis.

In each of the latter examples, one tends to fight two separate battles: the battle for an optimal mode of expression and the battle for an intuitive interface to an existing system. For example, consider a set of files and directories, which we want to have certain permissions. One has a notion of policy as a specification of the permission attributes of these files. Policy suggests that we should group items by their attributes. The existing system has its own idea of grouping structures: directories. A simple example of this is the following:

```
ACL1:                              ACL2:
 1. READ-WRITE /directory           1. READ-ONLY  /directory/file
 2. READ-ONLY  /directory/file      2. READ-WRITE /directory
```

Without clear semantics (e.g. first rule wins) there is now an ordering ambiguity. The two rules overlap in the specifically named "file", because we have used a description based on overriding the collection of objects implicitly in "directory".

In a real system, a directory grouping is the simplest way to refer to this collection of objects. However, this is not the correct classification of the attributes: there is a conflict of interest. How can we solve this kind of problem?

In the theory of system maintenance[Bur03], one builds up consistent and stable structures by imposing independent, atomic operations, satisfying certain constraints[CD01, Bur04b]. By making the building blocks primitive and having special properties, we ensure consistency. One would like a similar construction for all kinds of policy in human-computer management, so that stable relationships between different activities can be constructed without excessive ambiguity or analytical effort. This paper justifies such a formalism in a form that can be approached through a number of simplifications. It can be applied to network or host configuration, and it is proposed as a unifying paradigm for autonomous management with cfengine[Bur95].

10.4 Policy with autonomy

By a policy we mean the ability to assert arbitrary constraints of the behaviour of objects and agents in a system. The most general kind of system one can construct is a collection of objects, each with its own attributes, and each with its own policy. A policy can also be quite general: e.g. policy about behaviour, policy about configuration, or policy about interactions with others.

In a network of *autonomous* systems, an agent is only concerned with assertions about its own policy; no external agent can tell it what to do, without its consent. This is the crucial difference between autonomy and centralized management, and it will be the starting point here (imagine privately owned devices wandering around a shopping mall).

Assumption 3 (Autonomy) *No agent can force any other agent to accept or transmit information, alter its state, or otherwise change its behaviour.*

(An attempt by one agent to change the state of another might be regarded as a definition of an attack.) This scenario is both easier and harder to analyze than the conventional assumption of a system wide policy. It is easier, because it removes many possible causes of conflict and inconsistency. It is harder because one must then put back all of that complexity, by hand, to show how such individual agents can form collaborative structures, free of conflict.

The strategy in this paper is to decompose a system into its autonomous pieces and to describe the interactions fully, so that inconsistencies become explicit. In this way, we discover the emergent policy in the swarm of autonomous agents.

10.5 Agreements

In philosophy and economimcs, promises are often mixed together with the notions like contractsand agreements, and all considered to be related. In our theory of μ-promises, however, promises are a fundamental unit from which we can derive these other concepts.

10.6 Promise proposals and signing

Promises can exist and be discussed without them ever having been made or intended to be kept by an agent. We have used the notation $\psi(\pi)$ for a description of the promise π. This description must be adopted by an agent before it can be said to have made the promise. It is therefore useful to maintain the idea of promise proposals.

Definition 63 (Promise proposals) *The statement of a promise that is posited for consideration by one or more parties, prior to keeping or discarding the promise.*

Promise proposals are often discussed as part of treaty negotiations and commercial relationships. The terminology of these proposals is somewhat confused, so we define our own terminology below.

Definition 64 (Signing a proposal) *The concept of signing a proposal is to attach an agent's identity to is as a promise to use the proposal.*

10.6.1 The notion of an agreement

Not all agreements involve promises. An agreement is simply the arrival at a common conclusion by two individuals. For example, two parties can agree that $2 + 2 = 4$.

Definition 65 (Agreement) *A state of acceptance about a proposal that is shared between two or more parties.*

Two agents may or may not know about their common state of agreement.

Agreement about some body of information can be turned into a promise however if we phrase the acceptance as a promise, usually by signing. If all parties promise that a set of proposals will be honoured, then an agreement may be expressed as a promise to keep some specification or promise proposals. This may be called the *body* of the agreement. The term contract is also used here.

Definition 66 (Contract) *A bilateral bundle of promise proposals between two agents, that is intended to serve as the body of an agreement.*

Agreements are often assumed to be about policies or actions (agreements to act) and are often formalized using contracts in which case both parties agree to the terms of a common contract.

Definition 67 (Promise agreement) *A promise agreement is a pair of use-promises between two parties to acknowledge and adopt the body of a proposal.*

10.6.2 Legal agreements, terms and conditions

Many legal agreements may be considered sets of promises that define hypothetical boundary conditions for an interaction between parties or agents. The law is one such example. One expects the promises described in the terms and conditions of the contract will rarely need to be enforced, because one hopes that it will never reach such a point, however the terms define a standard response to such infractions. Legal contracts define what happens mainly at the edge states of the behaviour one hopes to ensue.

Note that a collection of promises is not necessarily consistent or sustainable. An agreement has to be worked out so that it is of mutual economic benefit to both parties, from each of their perspectives. This, however, has to do with the economic sustainability of the promises. An agent can clearly make a promise that it is in fact incapable of keeping.

10.6.3 Example: Service Level Agreements

Promises to keep within certain behavioural limits are clearly analogous to items that would be mentioned in Service Level Agreements, and hence we expect to be able to use them, within a suitable framework, to discuss Service Level Agreements. A promise, however, is not an agreement by itself. We must first explore the relationship between promises and agreements.

A service contract, for example, would be a contract from an agent that takes on the role of provider to another agent that has the role of client. The body of the agreement consists of the promised behaviours surrounding the delivery and consumption of a service,. From this we may define a Service Level Agreement as

Definition 68 (Service Level Agreement) *An agreement between two agents whose body describes a contract for service delivery and consumption.*

We can supplment this with defintions of the roles[BFa].

10.6.4 Cooperative agreement or treaty negotiation

If two agents agree to agree to behave according to the same standard as one another, then their common agreement is the intersection.

Definition 69 (Agreement)

$$\frac{a \xrightarrow{C(\pi_1)} b, b \xrightarrow{C(\pi_2)} a}{a \overset{\pi_1 \cap \pi_2}{\leftrightarrow} b} \tag{10.1}$$

We still have to decide what this actually means.

This clarifies the process and limitations of an agreement.. Each party might promise asymmetrically to keep certain behaviours, but the overlap is the only part on which they agree. Note that neither party can be subordinate to the other in such an agreement.

Notes

[32]It is a matter of belief whether one assigns this trust to a promise made by an agent called God.

11
Components and Modularity

11.1 Componentization of systems

The design of functional systems from components is a natural topic to analyze using promises: how we break down a problem into components, and how we put together new and unplanned systems from a pre-existing set of parts. To do this, we formally have to introduce the idea of design and purpose. The breakdown of a system into clusters of agents that make promises may be approached either from the bottom up or from the top down.

Decomposing systems into components is not *necessary* for systems to work. An alternative with similar cost-benefit is the use of repeatable patterns, but it is often desirable for economic reasons. A component design is really a commoditization of a reusable pattern, and the economics of component design are the same as the economics of mass production.

The electronics industry is a straightforward and well known example of component design and use. There we have seen how the first components: resistors, inductors, capacitors, and transistors were manufactured as ranges of separate atomic entities, each making a variety of promises about their properties. Later, as miniturization took hold, many of these were packaged into 'chips' that were more convenient, but less atomic components. This has further led to an evolution of the way in which devices are manufactured and repaired. It is more common now to replace an entire integrated unit than to solder in a new capacitor, for example. An analogy with medicine would be that one favours transplanting whole organs rather than repairing existing ones. This is all compatible with the agent abstraction, but it is worth spelling this out in some detail.

For example, in a particular context, an electrical appliance might be modelled as an agent that makes some kind of promise, perhaps to boil water or play music. However, we also know that it is built from smaller electrical compo-

nents that can be purchased separately, each making their own promises, and used to make any number of different circuits (coils, resistors, capacitors, etc). These could also be modelled as agents if we care for that level of detail. Electronic design is a showpiece for other industries in regard to component based design. Furniture manufacturers, currently exemplified by IKEA, have imitated this approach to sell furniture built from components like wood, screws, metal brackets etc.

Component based manufacturing is usually a bottom-up process. Top-down design does not usually work so well because it leads to components that are expensive to develop and are non-reusable. Most companies are not in a position to invent their own post office or delivery service, they would use the standard ones. Similarly, most companies do not drill their own custom designed oil and refine it, they buy fuel products from dedicated companies who standardize these things as off-the-shelf commodities. On the other hand, a furniture company like IKEA uses some re-usable components and some non-reusable ones to meet their requirements.

Let's try to describe the idea of components using μ-promises.

11.1.1 Definition of components

A component is an entity that behaves like a promise-agent and makes a number of promises to other agents within a given promise description. It can be represented as a bundle of promises. Since a component must fit together with other components, it is likely to be a parameterized object, so we may begin with the notion of parameterized bundles.

A component is normally a designed entity, so it is natural to discuss the role played by the component within the scope of a larger problem.

Definition 70 (Independent components) *An independent component is an agent that makes one or more unconditional promises.*

Definition 71 (Dependent components) *A dependent component is an agent that makes one or more conditional promises.*

In both cases, the definitions implicitly place the component within a larger system, since the promises must be for some other agents external to promiser.

11.1.2 What generic promises should components keep?

The kind of promise a component makes can be used to say something about its properties. In addition to the functional promises that a component would try to

keep, there are certain issues of design that would tend to make us call an agent a component of a larger system. Typical promises that components would be designed to keep could include:

- To be replacable.
- To be reusable.
- To behave predictably.
- To be self-contained, or avoid depending on other components to keep their promises.

What about the internal structure of agents in the promises? Clearly components can have as much private, internal structure as we like – that is a matter of design choice. Integrated circuitry, for instance, has been a major success in electronics. However, independence from internal structure can also be a strength. The more elementary the parts, the more ways there are to combine them into a creative chemistry. If we design components to rely on other components external to themselves, we haven't done anything fundamentally bad. But, if we design a component to have a mandatory dependence on a shared, exclusive resource then we have introduced contention. That increases the likelihood that the decision will lead to design problems in the long run.

11.1.3 Component design and roles

The simple structure of promises allows us to map components simply onto roles. A component is simply a collection of one or more agents that forms a role, either by association or a cooperative role.

11.1.4 Promise conflicts in component design

Components can make exclusive promises and thus they are also subject to the issue of promise conflicts (see section 3.13). It can be considered good practice to try to design for the avoidance of conflicts, however one often has incomplete information about the usage of components, making this impossible.

Definition 72 (Contentious components) *Components that make conditional promises, where the condition must be satisfied by an exclusive promise from another component may be called* contentious.

Contentious component design can sometimes save on costs, but it can lead to bottlenecks and resource issues down the line. Some examples of contentious promises include the following.

Example 16 *Multiple electrical appliances connected to the same power line contend for the promise of electrical power. If the maximum rating is a promise of, say 10 Amperes, they multiple appliances requiring a high current could easily exceed this.*

The foregoing example shows the effect of imperfect information during component design. The designer of an appliance cannot know the power availability when designing appliances, he or she must simply assume. A similar example can be made with traffic.

Example 17 *Cars could be called contentious components of traffic, as they share the limited road and petrol/gas pumps at fuelling stations.*

In the next example, a deliberate choice is made to save cost given the probability of needing to deliver on a promise of a resource.

Example 18 *Computer racks and 'blade chassis' are computers found in datacenters. Normally they do not have their own screens or keyboards, as these are rarely needed. A single keyboard and screen and DVD player might be shared between many of them. These blade computers may be called contentious as they expect to be able to use the common keyboard and screen at any time, even though this is not possible.*

It is worth remarking that contention is also closely related to coordination and calibration (see chapter 8), as a single point of coordination is also a single point of contention and indeed failure.

11.1.5 Reusability of components

Suppose we have a number of component agents that all make similar promises. For example, there could be a range of televisions to choose from, or normal and premium fuel when you visit the petrol/gas station, or even two brands of chocolate cake at the supermarket. All of these offerings can be considered components in a larger network of promises. How do we design these components so that other promises can be kept?

This is really a design question. Let us suppose our goal is to engineer components so that the promises they keep allow the components to be reused in different scenarios. There might be cases where it doesn't matter which of a

number of alternatives is chosen – the promises made are unspecific and the minor differences in how promises are kept do not affect a larger outcome. In other scenarios it could matter explicitly. For example:

- *Doesn't matter*: You would normally choose a cheap brand of cake, but it is temporarily unavailable, so you buy the more expensive premium version.
- *Matters*: One of the cakes contains nuts that you are allergic to, so it does not fulfill your needs.

For the widest reusability, we would normally try to avoid promise attributes that limited the likelihood of an acceptance promise. This mismatch of expectations reflects the basic promise axioms, that willingness to give does not imply willingness to use. In this example, the promise made by the cake was too unspecific for the user, who would not accept a cake with nuts. There was a mismatch between expectation and service, as the ill-fitting cake did not promise to not contain nuts.

Reusability implies that promises to use match the promises given in a *conditional promise binding*.

Definition 73 (Reusability) *Let $C_i, i = 1, 2, \ldots$ be a set of components that make conditional promises π_i with bodies $b_i|d$, where d is the body of a promise on which they all depend, delivered by component C_D. We can say that C_D is a reusable component if and only if $b_i \subset b(C_D) \forall i$.*

We can state this in more general terms.

Law 5 (Reusability law) *Components are reusable by other components if and only if the use-promise $(-)$ bodies of the components are at least as permissive than the give-promise $(+)$ bodies of all the service components.*

Notice that a component cannot be intrinsically reusable, but only reusable relative to the other components in an 'ecosystem'.

As an example of this, we can draw from software systems where versions of software sometimes fail to work together. Versioning or revision is an example of promise differences.

Example 19 *Suppose a computer has printer driver version 12 as a software component, and three software components use this version of the printer driver. The spreadsheet component only promises to print if version 13 or greater of the printer driver is available. The drawing software component only works with*

version 4 of the driver, or if a separate printing program is installed. However, a permissive calendar program has few requirements and will work with any, because it only uses a small number of the features of the driver.

In this example, one component has very specific requirements and is not fault tolerant. The second is picky about the version of the driver, but has a failsafe option to use a completely different piece of software to enable printing. The final component is easy going and is robust to the limitations of its dependencies.

11.1.6 Compatibility and interchangeability of components

When components change, or are revised, it affects the design of the entire cooperative system, and a total revision of the design might be in order. Replacement components are often said to require backwards compatibility. Let's examine what this means.

Definition 74 (Interchangeability of components) *A component is interchangeable with another if and only if it makes identical promises, i.e. it is merely an alias for another component.*

Example 20 *A knife can be interchanged with an indentical new knife in a restaurant without changing the function.*

Example 21 *A family pet cannot be interchanged with another family pet without changing the total family system.*

Compatibility of components is a weaker condition, that simply says it is possible for components to connect with one another.

Definition 75 (Compatibility of components) *A component is compatible with another if its $\pm$ promises can form bindings with the other components, and maintain the intended function.*

Example 22 *Antibodies recognize antigens like viruses because they have receptors that are compatible with the molecular signatures of the viruses*[33]*. We recognize smells because certain molecules are compatible with receptors in our noses.*

Example 23 *British power plugs are not compatible with the power sockets in America or the rest of Europe because they do not make complementary promises. Similarly, we can say that British and American power plugs are not interchangeable.*

Backwards compatibility is an issue often discussed when replacing components, especially software. Will replacing a component break something essential in a larger system? In fact the correct term is interchangeability as a new component might be compatible but still not function in the same way.

Often, when repairing devices, we are tempted to replace components with new ones that are almost the same, but this can lead to problems. Making all the same promises as an older component is a necessary condition for interchangeability, but it is not sufficient to guarantee no change in the total design. We need indentical promises – no more and no less, else there is the possibility that new promises introduced into a component will be able to conflict with the total system in such a way as to be misaligned with the goals of the total design.

Example 24 *Diesel fuel makes many of the same promises as petrol or gasolene, but these fuels are not interchangeable in current engines.*

Example 25 *When fingerprint readers were installed on to PCs in the beginning, they allowed users to choose between typing a password* or *using a fingerprint to enter the system. The upgraded keyboard added a promise which weakened security as allowing one or the other, rather than requiring both at the same time gave two possible ways of attacking the system instead of only one.*

Example 26 *When pluggable components were added to PCs during the 2000s, allowing upgrading to faster CPUs, the upgraded chips used more power and generated more heat. The power and cooling requirements thus changed and sometimes PCs would overheat.*

Example 27 *The insertion of new output transistors with better specification into a particular hifi amplifier actually improved the sound quality of the device. The transistors cannot be considered interchangeable, even though there was an improvement, because the promises they made were not the same, and the total system was affected.*

When upgrading or replacing components that are not indentical, a possibility is to design in a compatibility mode such that no new promises are added when used in this way. This allows component promises to remain unchanged and expectations to be maintained.

11.1.7 Specificity of promises and generic interfaces

Suppose we have a number of components that make similar promises. If the promises are identical, then the components are identical and it truly doesn't

matter which component we use. However, this assumes that the promises made by the agents are sufficiently specific and that there are no surprises in the way that the promise is kept. To make predictable components, there is an escalation game of promise specificity (see section 11.2.2).

Example 28 *Suppose a car rental company promises us a vehicle. It could give us a small model, a large model, a new model, a clapped-out wreck or even a motorbike. It could be German, Japanese, American, etc. Cleary the promise of 'some kind of vehicle' is not sufficiently specific for a user to form useful expectations.*

Example 29 *In a restaurant, it says 'meat and vegetables' on the menu. Does this mean a medium-rare beef steak with buttered carrots and green beans, or*

In both these cases, the promiser is making a 'lowest common denominator' or generic promise. The user, however, has certain standards to uphold and only promises to use a promise if the promise is specific enough.

This mismatch of expectation affects the way we make generic interfaces to similar promises. It is often desirable for users of promises to have a simple point of contact – sometimes called a 'front end' – that gives them a point of 'one stop shopping'. This has advantages and disadvantages to the user, as interchangability of the back-end promises is not assured.

Example 30 *Supermarkets often rebrand their goods with generic labels like 'megastore beans', or 'supersaver ice cream'. The lack of choice makes it easy for the user to choose, but it also means the user doesn't know what they are getting. Often in these cases, the promise given is implicitly one of a minimum level of quality.*

Example 31 *In a food court, the different food stalls offer meals for five dollars a piece woith hundreds of choices. You pay at the same checkout, but the stalls offer very different food.*

In a gourmet restaurant, the experience of choice is simlified to a 'taster menu', which the chef decides each night. There are some basic choices for special diets and allergies, but the simplification is based on the trust of the components designed by the chef.

Example 32 *In computing, a service provider offers a 'SQL database' service. There are many components that offer this service, with very different implementations. PostgreSQL, MySQL, MSSQL, Oracle, and several others all claim to promise an SQL database. However, the detailed promises are very different*

from one another so that the components are not interchangeable. Writing software using a common interface to these databases thus leads to using only the most basic common features.

When a promise giving component makes a generic promise interface to multiple components, it is forced to promise the *least common denominator* or minimal common functionality (see section 11.2.3). Thus generic interfaces lead to low certainty of outcome.

The user of a promise is the driver to decide what it needs to satisfy its requirements[34]. If the user promises to use any 'meat and vegetables' food promise by a caterer, he will doubtless end up with something of rather low quality, but if she only proimises to use 'seared filet, salted, no pepper', etc, the provider will have to promise to deliver on that, else she would not accept it to form a promise binding.

What we see is that a use-promise that is made to shop between alternatives needs to be sufficiently specific, and provide details of the requirements of the user agent. From the promiser component's perspective, there are two ways to address this differentiation:

- To make fully parameterized bundles of promises with many form-items to fill in.

  This is like ordering goods where detailed specification must be provided. This adds cost to the giver and overhead to the user, but allows total control by the user to drive what the giver promises. Essentially, parameterization of this type is a subordination promise (see section 6.4), because the component that gives service promises to give whatever the user asks for.

- To make a small number of alternative 'black-box' items, where the user selects between these unchangeable alternatives.

  This corresponds to limited product ranges, such as cars or supermarket goods.

Example 33 *A cabinet maker, making furniture to order offers a parameterized bundle of promises. Flat-packed furniture providers like IKEA offer predesigned alternatives without any parameterization.*

In car sales, we see both of these approaches used in a compromise. Components are designed in fixed models, but minor variations can be promised, such as colour, go-faster stripes, special brakes, etc.

11.1.8 Irrevocable component usage

Choosing a particular component implementation has long term consequences for component use and design. Sometimes a decision is made to select a particular non-interchangeable component. The differences that are specific to its promises then lead us down a path that moves farther and farther away from the alternatives because other promises will come to rely on this choice as dependencies.

Example 34 *Choosing to use a fuel type to power a car, e.g. using a car that makes diesel rather than a petrol/gasolene promise has irreversible consequences for further use of that component. Choosing the manufacturing material for an aircraft, or the building site for a hotel has irreversible consequences for their future use.*

Example 35 *Choosing to use a political party has few irreversible consequences, as it is unlikely that any other promises by the agent making the choice would depend on this choice.*

11.1.9 Interconnected components and design conficts

As discussed above, components can come into conflict if they make certain exclusive promises. This has an effect on the way in which we design a set of components, if they are going to be assembled in a way that makes one component dependent on another for keeping its promises.

This section on connection of components is directly related to the more generic discussions of dependency in sections 6.7 and 12.4, as well as the way intermediate agents, or the chaining of components ultimately leads to less certainty and even distortion of information in section3.6.

Consider figure 11.1, which shows how a single promise agent can depend on a number of other agents. If these represent components in a design, then each agent would be a replacable and upgradable part. How then can we be certain to keep consistent promises over time given that any of the components can be re-used multiple times and be interchanged for another during the course of a repair or upgrade.

Consider the component arrangement in figure 11.1, showing a component X that relies on another component Y, which in turn relies on component Z. The components exist in multiple versions, and the conditional promises made by X and Y specifically refer to these versions, as they must in order to be certain of what detailed promise is being kept (as in section 11.1.7).

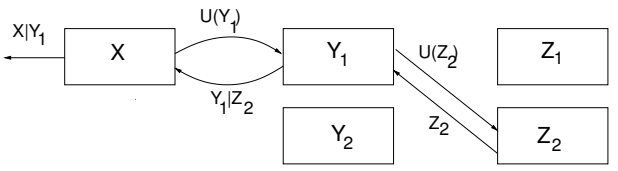


Fig. 11.1. A chain of component dependencies.

This situation is fragile if components Y and Z are altered in some way, there is no guarantee that X can continue to promise the same as before, thus its promise to its users is conditional on a given version of the components it relies on, and each change to a component should be named differently to distinguish these versions (see section 11.1.10 below).

To avoid this problem, we can make a number of rules that guide the design of components.

Rule 11 (Components dependency should be version specific) *Promises by one component to use another should refer to the specific versions that quench its promise to use. This will avoid later invalidation of the promises made by the dependent component as the dependencies evolve.*

Versioning takes care of the matter of component evolution and improvement in the presence of dependency. The next issue is what happens if a component is used multiple times within the same design. Can this lead to conflicts?

The properties of promises make component design straightforward. There are two issues surrounded repeated promises:

- If multiple components, with possibly several versions, make completely identical promises, this presents no problem, as identical promises are idempotent (see section 3.10).
- Incompatible promises, or exclusive promises prevent the re-use of components.

Rule 12 (Reusable components must avoid exclusive promises) *Any component that can be used multiple times in the same design must be able to exist along side other versions and patterns of usage. This means that components should not make conficting promises between versions.*

Example 36 *Suppose a castle has one flag for signalling. The guards promise to raise the flag if the king is at home, and lower it if he is absent. The courtier promises to raise the flag only on the king's birthday.*

Example 37 *The English word bow promises multiple meanings (it is a homonym). It means:*

- *An implement used to play instruments such as the violin*
- *To bend forward at the waist in respect (e.g. "bow down")*
- *The front of the ship (e.g. "bow and stern")*
- *The weapon which shoots arrows (e.g. "bow and arrow")*
- *A kind of knotted ribbon (e.g. bow on a present, a bowtie)*
- *To bend outward at the sides (e.g. a "bow-legged" cowboy)*

Most of these usages have distinct contexts and pronunciations. If a cowboy was ordered to bow, would he bend his legs outwards or double over?

The naming of components is a simple way to avoid some of these problems. We shall describe this in the next section.

11.1.10 The 'naming', 'identification' and 'branding' of components

The naming of components turns out to be a highly important issue for identifying function in a changing environment. Components need to promise to signal their function somehow, within the scope of possible users, else they would never be used. Naming allows components to be recognizable and distinguishable for reuse.

We must consider carefully what the naming components means, and use appropriate media. Perhaps 'identification of components' might be a better phrase, as names are usually thought of as things that are spoken or written. Since this is a book, we need a way to write about names, so we shall refer mainly to textual names here, but form and colour, tactile properties could be ways of creating a unique identity too. In marketing, this kind of naming is often called branding[35]. To avoid getting outside of the scope of a discussion about promises, we shall not try to desribe the variety of promises that identify a component, but rather focus on the structure of the promises that identify components uniquely.

It makes sense that bundles of promises be named according to their functional role, allowing components to be

- Quickly recognizable for use.
- Distinguishable from other components.

Naming needs to identify variation in time and space. Variation in space can be handled through scope, as it is parallel. However, variation in time is a serial property and therefore naming of a set of promises to be unique within a given scope.

Definition 76 (Naming or identity of component function (+ promises)) *A component name or identifier is any representative promisable attribute, rendered in any medium, that maps uniquely to a set of two promises:*

- *A description of the promises kept by the component (e.g. text like 'Ford Escort', or a shape like handle indicating carrying or opening)*
- *A description of model or version in a family or series of evolving designs (e.g. 1984 model, version 2.3, "panther")*

We keep a version identifier separate from the functional name because these things can be promises separately, and thus promise principles indicate that these are two separate promisers for the component.

Example 38 *In computer operating systems, software is usually componentized into 'packages'. These packages are named with a root identifier and a version number, e.g.*

```
flash-player-kde4-11.2.202.243-30.1.x86_64
unzip-6.00-14.1.2.x86_64
myspell-american-20100316-24.1.2.noarch
kwalletmanager-4.7.2-2.1.2.x86_64
```

The root name leads up to a number. Notice that the version numbering needs only to be unique for a given component, designated by the root name. The number then refers to temporal version, and a suffix `x86_64` *or* `noarch` *refers to version in 'space', or on different computer architectures.*

Within any particular scope, the user of components would normally see value in the uniqueness of names. We shall discuss this further in section 11.2. For now, we note that names need not be unique.

Definition 77 (Component alias) *Any component name that represents identical promises to another component may be considered a component alias for the other name.*

It is ultimately the responsibility of the user of a component to determine whether using a component's promise is justified or not. Components that lie about their

promises open themselves for non-cooperative 'tit for tat' behaviour (see section 15.5).

We may also decide to name the individual applications of a component, i.e. name a particular configuration in which a component is used. Usage need not be unique, as promises are idempotent so repeated use of a component would not make any difference to the formalism. In the physical world, however, a realization of a component can normally only be used once.

Definition 78 (Naming or identity of component usage (− promises)) *A description of what inputs are being supplied to the component (data, parameters) in order to use the promise components. A usage of a*

Attaching distinct names to instances of usage can be useful for identifying when and where a particular component was employed, e.g. when fault finding.

Example 39 *In computer programming, generic components are built into libraries that expose promises to using patterns of information. This is sometimes called an API or Application Programmer Interface. The application programmer, or promise user, makes use of the library components in a particular context for a particular purpose. Each usage of a particular component may thus be identified by the information passed to it, or the other components connected to it. We may give each such configuration a name 'authentication dialogue', 'comment dialogue', etc. We can also use the set of parameters given to an API interface promise as part of the unique name of the reference.*

There are definite advantages to having unique component names, i.e. a specific identifier like 'ipod touch' rather than a generic category name like 'music player'. Sure enough, we might want to distinguish different categories of object, but that is a different issue (see section 11.2.3). For now, we want to be able to attach a name to the unique set of promises offered by a component, so that we might make a catalogue of promises for designers to consult when selecting components[36]. We can always map a specific functional identifier onto a proper name, like swapping a serial number for the proper name 'the indigo 123XL super phone', however objects have names in many languages, so this is not so much about the specific words as the information represented by them[37].

If we want names to fulfil the promises above to be recognizable and distinguishable, then the names of components should be unique within a given scope. They must be considered shared resources.

Theorem 4 (Uniqueness of naming ± promises) *TODO.... From the perspective of μ-promises, unique naming of objects is necessary and sufficient to avoid promise conflict.*

PROOF

If names represent function, then they must also reflect changes to that function. Even if the goal of a particular component is constant (e.g. detergent should clean), the evolution of component design will normally lead to a succession of versions in an attempt to improve the components by various value-judgement criteria (see chapter 14).

11.1.11 Adapters, transducers and amplifiers

If there is no component available to quench a hungry conditional promise, it is sometimes possible to use an intermediary component to 'translate' the promises. Such a component is called a transducer or adapter.

Example 40 *An electrical appliance has been designed to conditionally promise functionality if it can use the electrical power. The componet was designed in the USA, but, in the UK, it cannot find a power point that promises the required voltage or pin configurations. Simply put, you can't plug the device into the power socket. The conditional promise made was too specific. However, it is possible to buy an adapter which promises to bridge the gap between what is available and what is required.*

$$\begin{array}{rcll} \text{UK socket} & \xrightarrow{square\ 3\ pin\ 240V} & \odot & \\ \text{US appliance} & \xrightarrow{function|flat\ 2\ pin\ 120V} & \odot & \\ \text{Adapter} & \xrightarrow{flat\ 2\ pin\ 120V|square\ 3\ pin\ 240V} & \text{US appliance} & \\ \text{Adapter} & \xrightarrow{U(square\ 3\ pin\ 240V)} & \text{UK socket} & (11.1) \end{array}$$

In the foregoing example, the problem for an electrical appliance could be made worse if there the power socket were an exclusive resource. If there is only a single component to satisfy the needs to many conditional promises, this must lead to contention (see section 3.13 for more details).

11.2 The economics of components

In section 11.1, we introduced the idea of components, or entities that behave like promise agents within a design context. Many devices are made up of components; a good example is the use of electronic components. Clothing is another example. We design special clothing-agents (shoes, hats, gloves, sweaters,

uniforms, etc) to fulfill different aspects of the promised function to insulate ourselves, and we combine them into a design, balancing functional and aesthetic values.

When does it make sense to build components? The answer usually lies in the economics of the promises involved. Components can lower the cost of certain operations. For example, it might be cheaper to replace a separable component than to replace a total system. Flat tyres would be a major inconvenience if one had to replace the entire car.

11.2.1 The cost of modularity

The cost of repairing a modular design is not necessarily a simple calculation.

Changing a tyre rather than replacing an entire car makes obvious sense, because the tyres are accessible and large.

Now think of electronic circuits. Components like transistors and resistors can be replaced individually, so it makes sense to change failed components individually, thus saving the cost of unnecessary additional replacements. However, the miniturizaton of electronics eventually changed this situation. At some point it became cheap to manufacture integrated circuits that were so small that changing individual components became impractical. Thus the economics of componentization moved towards large-scale integration of parameterizable devices.

When does componentization actually hurt you?

- When the cost of making and maintaining the components exceeds the cost of just building your own from scratch.
- If the components are not properly adapted to the task, because of special requirements.
- If the additional overhead of packaging and parameterization exceeds the saving

11.2.2 Choosing between alternative components: fitness for purpose

Suppose we have a number of components that make similar promises. If the promises are identical, then the components are identical and it truly doesn't matter which component we use. However, this assumes that the promises made by the agents are sufficiently specific and that there are no surprises in the way that the promise is kept. To make predictable components, there is an escalation game of promise specificity.

Example 41 *In sales, one often ends up selling every possible feature in an attempt to differentiate a promise and grab the attention of a buyer.*

The permissiveness of promise expectations referred to in the reusability law is an expression of *fault tolerance*, and this suggests another perspective on reusuability of components: quality and 'fitness for purpose'. Can we accept any old promise from a component dependency in a system?

Example 42 *Would we choose a postal delivery that took four weeks instead of four days? Would we accept balsa wood wings on an airliner instead of steel, or smaller screws that could handle smaller loads? Would we accept a less powerful computer in place of another to handle business transactions?*

All of these questions imply that there is a value associated with the promises made by different components, and that some promises are worth more than others. Naturally, this is why premium fuel costs more than standard, and hand-made luxury cakes cost more than mass-produced factory cakes. The evaluation of fitness for purpose is driven by the user and not the giver of a promise.

Marketing, camouflage and imitation are all strategies used to create deceptions around fitness for purpose[38].

11.2.3 Non-unique component promises and lowest common denominator promises

The value of unique names.

An agent always has incomplete information and hence cannot know if its name or identifying promises is unique. It would be extremely useful to designers if components had unique names, but this is simply not possible to arrange in all cases.

There are definite advantages to having unique component names, i.e. a specific identifier like 'ipod touch' rather than a generic category name like 'music player'. Sure enough, we might want to distinguish different categories of object, but that is a different issue

Brand identity

The user should decide what it needs to satisfy its requirements. If the user promises to use any 'meat and vegetables' food promise by a caterer, he or she will doubtless end up with something.

A use-promise that is made to shop between alternatives thus needs to be specific, and provide sufficient details of the requirements of the user agent.

Notes

[33]This is an over-simplification of the antibody mechanism, but the essence is accurate.

[34]This is like saying the customer is always right

[35]The concept of branding goes back to marks burnt onto livestock. In marketing, branding looks at many aspects of the psychology of naming, as well as how to attract the attention of onlookers to make certain comonents more attractive to users than others.

[36]In the Information Technology Infrastructure Library (ITIL), it is recommended to make a service catalogue of all business services. This is a form of componentization.

[37]In marketing, naming and 'brand identity' are major topics. In this case, the name of components (or 'products', as they would be called in marketing) would have values associated with them.

[38]In the theory of evolution, older books tend to refer to fitness, or the survival of the fittest. Their argument was that living systems that promise certain attributes fall short of what is expected to survive in an environment. These lifeforms thus exhibit the failure of some components to quench survival promises. For instance, I promise to run very fast if my heart will pump enough blood. The failure of a heart to pump enough blood will lead to the failure of the conditional promise. Of course, in evolution there is no conscious design taking place, merely selection through competitive failure.

12
Human-Machine Organizations

The concept of organization is loaded with many preconceptions and cultural associations. The word itself is used in two senses: organization as an entity (also called institution) and organization as a state of being (an orderly condition).

In this chapter, we consider how both of these usages can be given meaning in the framework of promises. We begin a discussion with the assumption that organization is a form of cooperative behaviour, in both these meanings, and that we can apply μ-promises to derive laws and characterizations that explain the behaviours and make predictions. Organization will emerge as persistent patterns in the trajectories of the system.

12.1 History

The Theory of Organization originates essentially with Max Weber.

The Wikipedia proposes that 'An organization is a social arrangement which pursues collective goals, controls its own performance, and has a boundary separating it from its environment.'

Weber, Max (1947) The Theory of Social and Economic Organization. Translated by A. M. Henderson & Talcott Parsons,The Free Press.

Actor-Network Theory..pretty close to promises, but no formalization?

Nearly all treatments of organization assume the doctrine of hierarchy in organizing agents, and never question the necessity of this assumption. We shall make no such assumption; however, we show how hierarchical structure emerges from economic and logical issues of cost and consistency.

12.2 Patterns of behaviour

As we have seen in chapter 7, behaviour is not explained solely by the promises made by agents; it also depends on environmental forces over which they have no control. Nevertheless, even in an unpredictable environment, systems may 'announce' their internal expectations by making promises to allow others to predict outcomes. Collective behaviour refers to behaviour exhibited by ensembles of agents. In this chapter, we consider the promises made by such ensembles of agents and discuss a framework for understanding how they can work in a coordinated manner. These elementary considerations form the basis for a discussion of cooperation and organizational patterns; it also provides the groundwork for a later definition of institutions and their boundaries[Ost90].

Behaviour is a pattern of observed change in the observable measures of an agent. When several agents are involved, we speak of collective behaviour. Behaviour is governed by the interplay between degrees of freedom and constraints (see chapter 7). Collective behaviour refers to a collection of agents that together form a behavioural pattern.

12.2.1 Patterns

Patterns can arise in two dimensions:

- Serial patterns of events in time.
- Spatial patterns of relatedness arising from the agents' promise ties. Clustering in the space of promise types - agents with similar promises are expected to behave similarly

Discrete patterns are described by the Chomsky hierarchy of grammars[LP97]. We may consider each distinct promise, or the operator that transmutes it into a persistent state to be a symbol in an alphabet Σ. The strings formed from this alphabet represent all possible behaviours in time, i.e. all sequences. The matrix of all such strings whose rows and columns represent the agents in an ensemble is also a matrix (a matrix of matrices) of some dimension. Patterns formed by reducible and irreducible blocks along the diagonal of this matrix represent patterns in the space of the agents (see fig. 12.3).

12.2.2 Order and disorder

We can characterize the variability of the trajectories resulting from promises.

- *Ordered behaviour:* An agent whose observable properties change according to a deterministic algorithmic pattern with a predictable grammar.
- *Disordered ("random") behaviour:* An agent whose behaviour changes in an unpredictable manner.

In general this is not a binary choice but a continuously varying scale, which is most naturally defined in terms of the entropy of the trajectory. We can define the order of an ensemble by

$$\text{Order} = \left(1 - \frac{S}{S_{\max}}\right) \tag{12.1}$$

where S is the Shannon entropy of a trajectory defined by

$$S = -\sum_i p(\hat{O}_i) \log p(\hat{O}_i), \tag{12.2}$$

and $p(\hat{O}_i)$ is the probability or normalized frequency of operator type i occurring in the time evolution of the behaviour.

12.3 Organizational roles

Roles are labels for agents or groups of agents derived from their observed or promised behaviour. The roles played by agents in an ensemble can be assigned simply by looking for all repeated patterns of promises. A role is thus a pattern. Since similar promises will lead to similar observed behaviour, we may define role labels for each distinct combination of promises that occurs in a promise graph.

- *Differentiated behaviour:*

  Agents that behave differently, e.g. perhaps partitioned into a division of labour when cooperating, or simple independent.

- *Undifferentiated behaviour:*

  Agents play identical roles in the ensemble and require no specific labels, since all promises are made by each agent.

Undifferentiated behaviour might be coincidental i.e. un-calibrated, like a disordered gaseous phase of matter (all the component elements coincidentally make identical unconditional promises but never interact with one another). Alternatively, agents might have agreed to behave alike through interaction (like in a solid phase of matter).

Normally we are only interested in the possibility of coordinated, collective phenomena, but a phase transition from one to the other is possible and we shall describe this elsewhere[Bal04].

12.4 Workflow logistics: go-betweens and intermediaries

Organization begins when agents choose to work together, so we shall study a simple model of cooperation: the logistic chain. Most business modelling assumes that organizations achieve logistical cooperation in a clockwork fashion, as if the organization were a single programmed unit. This implies certainty and subordination within the group that may or may not exist.

To understand the issues from a perspective of voluntarily cooperating agents, we begin with the simple case of a production-line chain, where there is causal ordering or a set of dependent prerequisites to delivering the promised result. Most systems are not systems of cogs and gears that make the outcome a direct result of an initial driving decision. The necessary causal ordering might nonetheless be realized simply through the occurrence of prerequisites, as explained by Hume[Qui98].

Let us begin with a simple chain. We use the language of promises, and the basic algebra of its conditionals, to examine how a 'controlling' or initiating agent can realize its promise through a number of intermediaries. For instance, suppose a business wants to make a promise to deliver a package a customer, typical intermediaries would be transport agents (networks, ships, the post office, etc.). This issue is often called the *end-to-end delivery problem*[Bura, Gao09].

12.4.1 Chain of promises

Since we cannot force one of these intermediary agents to behave perfectly, the agents involved need to secure a number of promises from one another. It is natural to ask what promises we then need to make to suggest a necessary and sufficient effort for the keeping of this 'distributed promise'.

Consider the following chain of promises.

$$A_3 \xrightarrow{b_3} A_2 \xrightarrow{b_2} A_1 \xrightarrow{b1} A_0 \ldots \tag{12.3}$$

As written, there is no indication in this graph that the agents are working together. Indeed, by assumption, the agents are all autonomous. A_3 makes a promise to A_2, but there is no indication here that this has anything to do with the completely independent promise to A_1.

Although we cannot force autonomous agents to work together, we can observe when there are sufficient promises made to conclude that they are indeed cooperating in the keeping of a promise, initiated by one of them.

12.4.2 Rewriting rule transformation

To understand promise chains it is useful to present a workflow as a succession of transformations starting from a single direct promise, and ending with an ensemble of agents acting as a chain of assistants. We therefore begin with a single autonomous promise by an agent that is solely reponsible for keeping the terms of the promise, and gradually add assistants that depend on one another in the manner of a chain.

Let A_i, R, $i = 0, \ldots, n$ be a collection of agents making promises with bodies $+b_i$, and let A_0 be the point of contact to overall recipient of the chain of promises b_0 to recipient R. In other words, at every stage, we are ultimately trying to keep a promise with body b_0 to R, with agent A_0 as the promiser.

$$A_0 \xrightarrow{+b_0} R. \tag{12.4}$$

This is unambiguous, however the addition of assistant will bring up a number of issues. Suppose that we now add a single assistant A_1, who promises A_0 body b_1 that it will rely on, in order to keep its promise to R. In accordance with rule 8 in section 6.3.1, we then have:

$$A_1 \xrightarrow{+b_1} A_0 \xrightarrow{+b_0|b_1\ ,\ -U(b_1)} R \tag{12.5}$$

The pattern above is a common occurrence, so we introduce a short hand notation for this:

$$A_0 \xrightarrow{+b_0|b_1,-U(b_1)} R \equiv A_0 \xrightarrow{b_0(b_1)} R. \tag{12.6}$$

This notation reflects the functional nature of the interaction between the agents. The agents are coupled by a binding or interface[BP06] of one $+$ and one $-$ promise.

The introduction an assistant suggests a re-writing rule that we can repeat for additional assistants. Let us now add A_2 to assist A_1 in the delivery of its promise to A_0, by iterating:

$$A_2 \xrightarrow{+b_2} A_1 \xrightarrow{+b_1(b_2)} A_0 \tag{12.7}$$

Thus we have made the replacement $b_1(b_2)$ for b_1, which means that we should also make this replacement in the use-promise so that $b_1(b_2)$ is used by A_0. The

resulting chain is now:

$$A_2 \xrightarrow{+b_2} A_1 \xrightarrow{+b_1(b_2)} A_0 \xrightarrow{b_0(b_1(b_2))} R. \tag{12.8}$$

This simple substitution seems straightforward, but there are two problems with the result. First, there is an apparent ambiguity in the meaning of a 'promise to use' of a conditional promise that we have to resolve; and, second, A_0 now refers explicitly to a promise with body b_2 that it has not personally been promised.

12.4.3 Naive rewriting ambiguity

Consider first the meaning of the chain-link binding $b_n(b_{n+1}(\cdots))$. From our definition in eqn. 12.6,

$$b_0(b_1(b_2)) \equiv b_0|b_1(b_2)\ ,\ -U(b_1(b_2)). \tag{12.9}$$

Hereafter we shall drop the $\pm$ symbols for brevity.

Let's dispense first with the conditional part of this rule $b_0|b_1(b_2)$. Making a body conditional on a conditional brings straightforward recursion. If we try to recduce this, this suggests:

$$b_0|b_1(b_2) \equiv b_0|b_1|b_2\ ,\ U(b_2), \tag{12.10}$$

Since the body types are independent, there are no particular ordering issues between the use promise and the conditional order. In total then, we seem to have a re-writing rule of the form:

$$\frac{b_0(b_1(b_2))}{b_0|b_1|b_2\ ,\ U(b_1(b_2))\ ,\ U(b_2)} \quad ?\ (\text{no}) \tag{12.11}$$

In this expression, we note the appearance of $U(b_2)$; this refers to a promise with body b_2 that was not made to the agent, but was passed on as 'hearsay'. However A_0 clearly cannot promise to use a promise made to another agent, so we have to decide on the meaning of this – of a use-promise about a $+$-promise, whose body b_1, which is itself conditional on a different promise with body b_2, that A_0 has not actually been given.

We shall discuss two interpretations and show that it is reasonable to view these as being equivalent up to a level of *trust* in the integrity of knowledge about the process.

12.4.4 Scope interpretation of assisted promises

The ambiguity in interpretation arises from a weakness in the notation. As written, the use-promise notation is incapable of reflecting the fact that the promise

we are using was in fact made to another agent. The 'usage' of promises to others makes no clear sense. The same argument does not apply to the conditional part of the promise, where we actually have a conditional chain $b_0|b_1|b_2$ without ambiguity.

To resolve this interpretation of $b_0(b_1(b_2))$, we need to bring back the issue of *scope*, or knowledge about promises. In section 6.4 we wrote the knowledge of a promise using the notation $\psi(\pi)$ or $\psi(\pi(b))$. What we see here is that scope is extended implicitly by communication of conditional information, as long as we trust the integrity of the intermediate agents (see section 3.6.5).

The promise our agent is trying to 'use', according to this slavish application of the re-writing rule, is not the actual promise to another agent itself, but rather *knowledge* of the promise $\psi(\pi)$. Thus the first interpretation of this is to make a rule ignoring nested use promises, so that each agent simply trusts the integrity of promises that are asserted through conditionals, even when referencing far-away agents. Thus, what we are really claiming is this:

$$\frac{b_0(b_1(b_2))}{b_0|b_1|b_2\ ,\ U(b_1(b_2))\ ,\ U(\psi(b_2))} \tag{12.12}$$

as whatever else $U(b_2)$ promises it will gobble, nothing more is being promised by an agent in the chain.

The first resolution is a simple approach, and it suggests that scope is extended naturally by virtue of the conditional, so further information is redundant. However, it does ignore the lack of integrity of the information. We only have the nearest agent's word for it, that all the necessary promises will all be used.

The second resolution is to note that there seem to be a number of use-promises in eqn. 12.12 for which there are no $+$-promises. The promise $U(\psi(b_i))$ that we obtain through iteration, is left hanging – it goes 'hungry' for want of a direct promise of type $+\psi(b_i)$ from its source.

But this is a strong suggestion. Our agent needs to have direct confirmation that the assistant agents are going to keep all promises along the chain that will eventually impact on *its* conditional promise, so we could arrange a extension to the re-writing rule to quench each one of these use-promises, forming a definite binding. The implications of this however are considerable. It requires a total number of promises of the order $O(N^2)$ for a chain of length N, in other words, a complete graph[39]. This is certainty at a high price.

Let us summarize these resolutions:

1. We argue that the $U(\psi(\pi_i))$ promise is effectively and automatically quenched by the promise body referring to b_i, since the rewriting rule 8 insists that

confirmation of usage be passed along the chain. The only difference between these is that the agents do not have direct confirmation that the assistants will keep their promises, only hearsay. The certainty is impaired with distance, but the cost in terms of promises is only order $O(N)$.

2. We argue that knowledge of some condition $\ldots|b_i$ is not the same as a direct promise $+\psi(\pi_i)$ about keeping π_i to our agent. A_n must actually receive this promise to have maximal certainty that the conditional promise is intended in the way it is described. The implication is that every agent for $i > n$ would have to promise A_n knowledge of their promises to A_{i-1}, all along the chain, thus quenching $U(\psi(b_i))$. We end up with a total 'cost' for the workflow process at $O(N^2)$.

These two solutions are indicative of all graph architectures. There is an $O(N)$ solution and an $O(N^2)$ solution, e.g. centralized hub versus direct connection. We should not be surprised by these outcomes. In each case there is a cost trade-off between certainty and cost in terms of keeping promises (in this case an information cost).

12.5 Formation of hierarchy

Hierarchy lies behind practically all visions of what constitutes systematic and organized behaviour in the literature. We propose in line with many previous authors' thinking that hierarchies emerge for microeconomic reasons. We shall diverge from other authors in the suggestion that this is inevitable. Each agent, after all, pursues its own interests autonomously and the resulting collective behaviour reflects an evolutionary process[Axe97, Axe84]. Despite its widespread dominance empirically, we resist the *assumption* of hierarchy as a pattern of organization.

The characteristic of hierarchy is the existence of a root node, or privileged agent at the top. the question is how this node gets selected from a group of agents. A full understanding of this phenomenon requires a discussion of symmetry-breaking, however we can discuss a simplified view. We suggest that this emergence of privilege has a simple explanation in a process of structural 'crystallization' which is seeded by a self-appointed promiser of a collation service. The economic advantage to the leaves for a privileged observer is that can make comparisons within the initially flat ensemble more cheaply than having each individual agent establish peer-to-peer communication with every other agent in the ensemble. The cost benefit of such centralization depends on how many promises need to be set up and maintained.

There are two separate economic issues in ensembles: the cost of *calibration* or the attainment of global measures allowing consistency, and the cost of *coordination* or differentiation and delegation which requires only local consistency. Calibration requires complete bi-directional communication between all agents. This is a familiar problem in security where it is used for key distribution[Bis02]. Coordination requires only that we can pass a message to every agent on a need to know basis, without the necessity for reply. Without calibration, agents have only local concerns and global ones are considered to 'emerge' i.e. they are un-calibrated (we return to emergent behaviour below).

The local economics of network relationships are quite simple and depend mainly of the topology at a point. We need to show how the cost of a particular topology impinges on the cost of either coordination or calibration. We should recall that promises are not about continuous network communications, so the cost of making a promise is entirely in the establishment of the promises. The maintenance of the promise depends on its type however. Promises that require an exchange of information between agents involve propagation of data, which introduces measures of time-taken, latency etc.

Promise graphs form networks and the economics of coordination thus have two facets (see fig 12.1): cost and efficiency. The cost of establishing promises increases with the number of promises since each promise generally requires some behaviour or work to be done by the agent. The efficiency of coordination involves communication and therefore has to do with propagation of effect over the coordination distance (this is a network depth issue). We can divide the discussion into what is good for the group and what is good for the individual agent.

12.5.1 Global considerations

There are two extreme cases for topological connectivity in a global region and a range of values in between. These are the complete graph (all pairs nodes linked peer to peer with $N(N-1)$ directed links) and that of centralized hub (with $(N-1)$ nodes linked directly to a single hub, making $2(N-1)$ directed links). If we assume that, to a first approximation, agents are homogeneous and value promises from one another equally then the cost and value of promises is proportional to the number of promises.

If agents do not route messages to each other through some preorganized structure (such as a spanning tree requiring many coordinated promises), they have to coordinate with every other agent individually in a complete graph of $N(N-1)$ promises of each type of promise in the ensemble of size N. If they

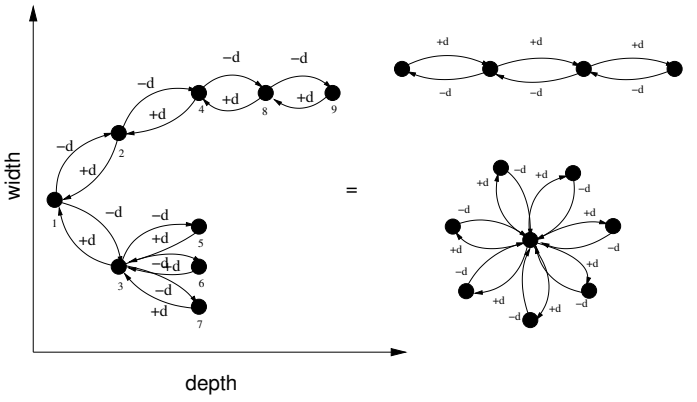


Fig. 12.1. Bilateral communication structures indicating "depth" and "width" of promise bindings. A tree is something between the extremes of a chain and a hub.

network their efforts however into a hub or chain then they can reduce their promises to order $2(N-1)$ in total, but now there is a new issue: depth or efficiency.

12.5.2 The depth versus width conundrum

The problem of coordinating in a tree structure or hierarchy has two competing costs: the cost of coordinating with peers at the same level (sideways or breadth search), and the cost of searching sub-layers descending or ascending the levels of the hierarchy in search of the destination (depth search).

Depth versus width is a trade-off. Centralization at a hub reduces depth and hence increases the coordination efficiency, but it increases the cost burden of promises at the hub. The costs are in-homogeneously distributed. In a chain (the opposite of a hub), the cost of keeping promises is maximally distributed but the depth is maximal too, meaning low coordination efficiency and delays.

12.5.3 Local considerations

Agents do not generally see the global picture; they care only about their own costs and benefits. This means that the true picture of cooperative behaviour will necessarily be inhomogeneous in all cases but a complete graph, which all agents will perceive to be expensive for large N. The difference between these must be $N(N-1) - 2(N-1) = (N-1)(N-2) \gg 0$ for justify

appointment of a privileged collator. As soon as the privileged collator has been chosen, the cost to non-privileged agents is simply 2. However, as N grows, the cost for the appointed collator grows linearly like $2(N-1)$. One solution is to look for a balanced tree, like a Cayley tree[AB02] which allow constant scaling of promise cost for all agents, however in this case the depth of the structure increases, leading to a rapid fall-off in efficiency, thus there is a trade-off.

Optimizing the structure is a simple matter of comparing the relative economic merits of these two properties. Let k be the average node degree for promises in a tree. The cost associated with not getting data quickly is proportional to the effective depth of the network pattern $(N-1)/k$, then we have a cost function that is a balance between these two. All cost functions contain arbitrary (subjective) parameters. In this case we denote ours α:

$$\text{Cost} \propto v_i \left(\bigoplus_i (a_i \xrightarrow{-d_i} a_{i-1}) \right) = \alpha \left(k_i^{(-d)} \right)^2 + \frac{(N-1)}{k_i^{(-d)}}. \quad (12.13)$$

A plot for this for the arbitrary policy $\alpha = 0.1$ is shown below. This shows the existence of an optimum aggregation degree, in this example $k = 5$. Such arguments should also be taken into account in the scaling argument, as we see the cost rise sharply with increasing centralization. This shows a plausible

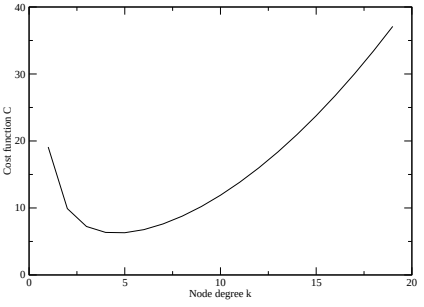


Fig. 12.2. Cost considerations can plausibly lead to an optimum depth of network pattern when power considerations are taken into account. The minimum cost here is given for $k = 5$. Such considerations require an arbitrary choice to be made about relative importance of factors.

explanation for why a hierarchy emerges. It seems locally cheaper per agent than a full mesh and it can tune its efficiency as long as the structure does not become fixed.

There is a paradox here: agents break symmetry to appoint a leader in order to cheaply scale the number of interactions required to compare and calibrate

outcomes from multiple agents for the "client" agents, however the appointed agent ends up choking on this burden eventually. The solution that generally emerges is a kind of lazy-evaluation: agents do not make promises that they do not need to make. What then emerges is often something like a small-worlds network or power-law structure[Wat99, NSW01, Bar02] which is seen in peer to peer networks. This suggests that ordered management is not something that scales without active abstention from coordination.

12.6 Organization from differentiation

Let us now examine the word 'organization' and try to provide a definition below that is unambiguous. How does *organization* differ from *order*, for instance? In natural science, self-organization generally means spontaneous differentiation or clustering, i.e. a reduction in local entropy of an open system.

Is a tree considered organized, or merely ordered? The now established term self-organization forces us to define the meaning of organization clearly, since it implies that organization may be something that is both identified *a priori* by design, or *a postiori* as a system property.

Intuitively we think of organization to mean the tidy deployment of resources into a structural pattern. "Organization" (from the Greek word for tool or instrument) implies to us a tidy compartmentalization of functions. We know that all discrete combinatoric patterns are classified by grammars of the Chomsky hierarchy[LP97, Bur04a], which may be formed from the alphabet of such operators. This is consistent with the concept of lowered entropy and differentiation. Organization requires distinguishability.

Patterns may be formed over different degrees of freedom; for instance:

- Spatial or role-based partitioning of operations between parallel agents.
- Temporal (schedule), i.e. serial ordering of operations at an agent.

For some, an organization also imbues a conscious decision amongst a number of agents to work together, with a hierarchical structure, and a leader, e.g. with a *separation of concerns*, or *division of labour* in the solution of a *task*. Many also believe in the value of *re-usability* (a subjective valuation of implementation which could lead to an economic criterion for selection of one structure over another).

We prefer to think that all of these can be understood economically. Two agents trained to fight a fire could both independently promise to grab the fire extinguisher or dial 911, but if they promise to divide the tasks then both tasks

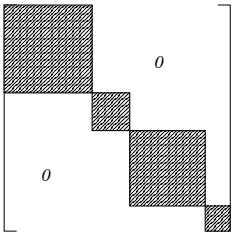


Fig. 12.3. Patterns in agent space arise from irreducibility of the graph.

will be started sooner and finished earlier costing less totally and improving efficiency by parallelism.

Parallel efficiency gain is the seed for *differentiation*; its survival is a matter of sustained advantage, which requires sustained environmental conditions.

12.7 Organizations or institutions

We now define *an organization* or *institution* as a discrete pattern that is formed from interacting agents and which facilitates the achievement of a desired trajectory or task, i.e. a change from an initial state $\vec{\mathcal{Q}}_{\rm i}$ to a final state $\vec{\mathcal{Q}}_{\rm f}$ over a certain span of time. We refer to the discussion of systems in ref. [Bur03] for the definition of a task.

Definition 79 (Organization) *A phenomenon in which a pattern forms in the behaviour of an ensemble of differentiated agents.*

Let $\mathcal{E}$ be an ensemble of distinguishable agents. The observables of $\mathcal{E}$ can formally be written by direct sum $\vec{\mathcal{Q}} = \vec{q}_1 \oplus \vec{q}_2 \oplus \ldots \vec{q}_N$, but we do not assume that these are public knowledge to an actual agent. An organization over the ensemble consists of the tuplet $\mathcal{Z} = \langle \mathcal{E}, \mathcal{Q}, \mathcal{A}, \mathcal{S} \rangle$, where: $\mathcal{E}$ is a set of agents with a promise graph $\mathcal{A}_{ij}$. $\mathcal{S}$ is a string of matrix operators for the whole ensemble $\hat{\mathcal{O}}_A(a_i \xrightarrow{\pm *} a_j)$ which describes the observable changes made by agents, for some sequence index A, Diagonal elements of $\mathcal{S}$ include the operations $\hat{O}_A(t, \vec{\sigma})$. $\mathcal{S}$ spans all the observables in the ensemble with column dimension $\sum_{i \in \mathcal{E}} \dim(\vec{q}_i)$, and modifies the observables of all agents: $\hat{\mathcal{O}}\vec{\mathcal{Q}} = \vec{\mathcal{Q}}'$.

Organization can now be understood as a discrete pattern induced with $\mathcal{Z}$. We discern orthogonal types of organization (analogous to the longitudinal and transverse nature of wave patterns in a continuum description):

- **Serial organization** is the syntax of operational changes $\mathcal{S}$, classified by a Chomsky grammar.
- **Parallel organization**: is the partitioning of $\mathcal{Q}$ induced by the irreducible modules of $\hat{\mathcal{O}}_A$ at each serial step. This is a property of multiple agents and is characterized by the eigenstructure of $\hat{\mathcal{O}}_A$, which defines natural regions in the graph[CEM04].

What of 'an organization' as a noun (e.g. an institution or company)? We normally think of this as a number of actors who are organized under the umbrella of some architectural edifice – like a building. Is it enough to simply collect agents within a boundary to make them an organization? We think not. Our definition above still works in this case, but it does not quite fit the facts. An organization is clearly an ensemble with collective behaviour, and it clearly forms a pattern (even a trivial one); however, organizations or institutions as we understand them always have boundaries (which one may or may not consider artificial).

The only natural boundary for interaction is to limit our understanding of organization to the point where no more promises are made. However, this would mean that every business had all of its clients as part of its organization, which is not our common understanding of what an organization is. Where is the edge of a pattern? The resolution lies in the common usage of organization as a synonym for institution. If an organization could have a boundary it would be completely isolated from other agents, a breach in a boundary (a leaky boundary in the parlance above) would demand that we extend the boundary to include the part it is interacting with. This in turn means that the boundary is not a boundary. Any ad hoc definition of the edge of the organization (the edge of the pattern) would be arbitrary and subjective, and no agent would be able to know whether it were inside or outside the organization itself. However, therein lies the clue. How would an agent know? The boundary is not defined by interaction but by a specific promise type, by a promise of membership. The common meaning of an organization is as follows:

Definition 80 ('An' organization) *A number of agents that each promises to be identified as members of an organization.*

Organizations are thus more than a pattern that identifies 'collective agents' making promises that a single agent would not be able to make alone. They are also self-appointed *roles*.

12.8 Empirical support for promises

There are ample studies to in the published literature in which to seek validation of at least some of the foregoing ideas. We propose to narrow our focus to just a few of these, since a full treatment would warrant a major study that is beyond the scope of the present paper.

The first part of our paper concerns laws of transmitted influence. The three laws themselves are proven from axiom and therefore the only validation required is of the assumptions on which they are based. Since the assumption is of autonomy, and one can always model a non-autonomous agent by an autonomous one armed with promises of submission, there is nothing worthy of verification. The laws are simply expressions of necessity, and we may turn the argument around to predict the existence of effective promises in all cases where influence is transmitted between components. Such promises are often represented as access control rules and network services. We encourage readers to be on the look-out for such promises in systems that seem to be under the control in a master-slave relationship. We are satisfied that they are present.

The importance of promises as opposed to events is their relative persistence, i.e. in allowing us to understand the stable properties of behaviour rather than passing transitions. A promise's outcome is represented by a distribution of outcomes rather than a single response to an event. Distributions have greater stability than individual observations. This is where the promise model differs from other works on the Event Condition Action Model. We expect to find predictability only at a statistical level, and have previously found this to be the case[Bur00].

Apart from these minor verifications, we find most support from indirect studies of organization[Fox81, ZM03, SBMR07]. This is no doubt a result of contemporary predominance of interest in networks and their management. In this, two areas distinguish themselves for their clear dependence on *calibration* and *autonomous promises*: the Border Gateway Protocol and encryption key verification. Both of these subjects have been studied at length, especially the former. The data from these works are most useful in supporting ideas about organization and structural crystallization from peer-promises to centralized or hierarchical structures.

BGP studies are particularly interesting not for their routing at the level of packet events but because BGP behaviour has long term trends that are based on policies given by autonomous systems (AS). BGP policies are clearly promises by our definition concerning the transit of packets. Two phenomena are of interest: transit services and peering. Norton was amongst the first to consider the

habits of BGP users[Nor01a, Nor01b] and his result support the conclusion that the bindings made between AS's have little to do with packet traffic or transit tariffs, but rather everything to do with the potential value of their promises to peer with other powerful providers – in terms of social capital. Only when the cost of keeping these promises becomes debilitating do service providers waver from these promises. This is an explicit example of the importance of promises over events.

BGP also allows us to see delegated address spaces[SBMR07], which in promise terms implies a growth of autonomous agents and promises between them. The resulting structure of collaboration for sharing of the environmental resource is an organizational pattern. Sriraman et al. show that the structural organization of this is hierarchical from the top down. This kind of top-down phenomenon is not covered in our work because it occurs when a single agent splits into several agents with promises that link them to the residual of the original agent. Thus inevitably leads to a local cluster attached to an anchor point, but what is interesting is that the economics again drive the formation of a basically homogeneous tree in accordance with our predictions. The node degree of the graph is remarkably homogeneous, suggesting that a fixed number of promises is reached by a balance akin to that of eqn. (12.13).

Zhou et al.[ZM03] make the point that this homogeneity is only a local phenomenon. The actual degree distribution of the BGP network follows a power-law behaviour with a long tail[NSW01, Bar02]. Its average node degree is about 6, but maxima of up to 3000 are found. As we have pointed out, it makes sense for all but the richest resource providers to keep their promise counts low relative to their own capabilities – and these are not homogeneous. The most convincing support for our model comes from Norton's interpretation of the value to providers in bilateral peering[Nor01b]. He shows that the perceived value of outsourcing promises for major providers is a privately measured value and grows with increased peering relationships up to a maximum limit at which point it tails off, throttled by the resource bottleneck of the hub. This is mirrored in Dunbar's theory of human peering in anthropology[Dun96].

A second area of computing in which organization structure is linked to economics of promises is in encryption key management. Here there are two basic models: direct key exchange, such as is used by tools like Secure Shell, and the trusted third party broker used by Transmission Layer Security SSL/TLS and Kerberos[Bis02].

Dondeti et al. [DMS99] provide some evidence to suggest that the cost distribution of key verification promises is already quite uniform, suggesting that the centralization bottleneck has been outpaced by improvements in technology.

Clearly affordance can be sated either by a resource "arms race" or by diffusion of load.

The body of literature from economic organization theory is derived not only from economic game theory, but also from the observations made about organizations throughout its rather longer history, thus it brings a more complete and less artificial verification of promise predictions. A compelling survey of these ideas was found in the work of Fox [Fox81], whose main points bear a striking resemblance to our results and therefore indirectly validate them: ours are based on simple axiomatic theory with few assumptions but predictive power, whereas his are based on experience of actual organizations. Human influence is prevalent in computer behaviour, since behind every computer system there lies a human decision-maker, vying for the value and success of the system.

If ref. [MR72] the authors define organizations as collaborative structures: "a group of persons who actions (decisions) agree with certain rules that further their common interests". Further they define teams: "an organization whose members have only common interests". We find these definitions typical of those in economic theory, and motivated more by wishful thinking than on the basis of an impartial model. To begin with, they are not founded on elementary concepts. Promise theory shows that cooperation is not an elementary concept but in fact requires a plethora of promises to accomplish. We have uncovered a deeper understanding based on simple arguments that both confirms prior experience and enhances it with mainly the assumption of autonomy at the heart.

There is much scope for future work in understanding the empirical verification of promise theory. Empiricism lies at its heart, so this is no small challenge. The magnitude of the task is also an indication of its actual substance.

12.9 Conclusions

We have shown that behaviour is a pattern of change that can be partially predicted by reducing a system to an ensemble of autonomous agents making promises. Three basic laws of influence on behaviour follow from the property of autonomy that underlies promise theory. These laws are elementary expressions of change, explaining the promises required to transmit influence between autonomous agents; thus they describe the meaning of transmitted "force".

Observation is the cornerstone to understanding agent behaviour because promises never imply guaranteed determinism, only distributions of outcomes that can be observed in experimental trials. Our ability to distinguish agents stems from our ability to distinguish their behaviour, either in advance (from promises) or after the fact (from their observed trajectories). The ability to observe differences in

behaviour is not guaranteed: there are symmetries in ensembles of indistinguishable agents that require the calibration scale of a single adjudicating observer to gauge. Once a scale of distinctions can be made, the concept of organization can be explained empirically, purely in terms of observable patterns of variation, without the need to imagine that they are always the result of complex human concepts like 'designs' or 'goals'. The 'self-organization' is a redundant terminology, as organization is a measurable property of any system.

A frequently emerging pattern is the hierarchy. We argue that this does not emerge out of the need for control or separation of concerns as do other authors, but rather from the avoidance of economic cost associated with observing and distinguishing system components on a single calibrated scale of measurement: the comparison of capabilities.

The descriptions we offer here are a platform from which to clarify many issues in autonomic systems. There are unanswered questions about the subjective nature of agent perceptions that motivate the need for a full theory of measurement based on promise agents. This work is only the beginning. From such a theory it should be possible to decide whether peer to peer and centralized systems are comparable organizations with interchangeable properties, or whether they are two fundamentally different things [40].

Notes

[39] A complete graph is one in which every agent is directly connected to every other agent by a private connection, i.e. it has the maximum number of $N(N-1)$ directed links.

[40] We believe that it is possible to go further and define material properties for promise graphs, by analogy to how physics describes the large scale properties of matter from an atomic model. Why is wood strong and glass brittle? Why is one computational structure robust and another fragile? These are analogous questions that are about scale as well as the underlying promises that bind the parts into a whole. We must work towards suitable and useful definitions of these properties. We believe that this such definitions must follow from promise theory or something like it. We return to these issues in future work.

13
CFEngine: A Promise keeping engine

CFEngine is a software tool that automates the building and maintenance of networks of computers at the software level. it assures the configurations of computers: installing and customizing their software for specific purposes, hardening their settings against security threats, ensuring that only the right programs are running, and a variety of other administrative tasks associated with computer management.

The story of promises, as we describe it in this book, has been intimately related to CFEngine. in fact the importance of promises was first realized in 1994 while trying to find a model that could describe CFEngine's mode of operation. CFEngine uses a model of completely decentralized and distributed management, but most management frameworks that existed at the time of its inception were fully centralized, authoritative remote-control consoles.

With CFEngine running, every computer becomes responsible for its own 'state of health', and cooperates loosely with neighbouring computers if and when necessary. the term 'computer immune system' was used, at one time, to visualize this form of self-sufficiency. CFEngine solves a problem that centralized systems cannot easily solve, by providing a massively scalable parallel system of configuration, and thus it offers a strong case study for the concepts in this book.

In practial terms, CFEngine combines the notions of promises with the ability to represent patterns (grammars) of an alphabet of promises. in this way, it models configurations that can be promised and kept autonomically.

13.1 Policy with autonomy

CFEngine implements computer policy. by a policy we mean the ability to assert constraints on the behavior of objects and agents in a computer system. the most general kind of system one can construct is a collection of objects, each with its own attributes, and each with its own policy. a policy can also be quite general: e.g.

- Policy about behavior,
- Policy about configuration, or
- Policy about interactions with others.

With a promise model, and a network of *autonomous* systems, each agent is only concerned with assertions about its own policy; no external agent can tell it what to do, without its consent. this is the crucial difference between autonomy and authoritative centralized management. autonomy means that no agent can force any other agent to accept or transmit information, alter its state, or otherwise change its behavior.

13.2 History

Before 2004 CFEngine, like most computer programs, had been written based mainly on intuition of the problem it was trying to solve – namely computer configuration in the highly diverse environment of a university, where many users had special needs to be taken care of. however, after many years of ad hoc experimentation, it became necessary to formalize a model for distributed resource configuration to bring order to this irregular intuition. curiously, no such suitable model existed in computer science, as distributed resource management is not something that has been modelled traditionally. computer science tends to treat problems as task graphs, or computer programs to be executed, not as landscapes of multifaceted properties to be described. promises were the answer to unifying many of the concepts in CFEngine.

The promise theory described in this book was originally invented to discuss the issues surrounding this autonomous operation, and voluntary cooperation. unlike other modeling techniques, like Petri nets or UML, promise theory is not about the stepwise development of a device. it is not about protocol modeling, rather it is about *equilibria*, i.e. how to describe steady state behavior that has some underlying dynamics.

Promise theory begins with the idea of completely autonomous agents that interact through the promises they make to one another. it is therefore particularly well-suited to modeling CFEngine.

CFEngine maintains the principle of *host autonomy*. many existing systems and descriptions of configuration management assume a centralized authority to change all the parts of the system. this has a number of disadvantages. our aim here is to begin in the opposite manner, assuming *no centralization* of authority, and then seeing how collaborative structures emerge by free choice.

Promise theory applies to CFEngine at a number of levels:

- in the way the software is designed.
- in the way parts of the software cooperate.
- in the model used by the software to describe the system.

To apply a promise method to any of these areas, we observe the basic principles and assumptions of the theory. applying promise theory to a representation of promise theory is a strangely self-referential thing to do, but it is the only way to be consistent about the principle. it also shows that the idea of promises can be applied recursively at multiple levels in a description of function.

We must begin by identifying the atomic 'agents' in the software along with the promises they make, as well as the atomic agents that are described within the software – i.e. in its model of a computer system through an internal representation of promise statements. put simply, this involves the following points:

1. identify as many of the resources and their properties, i.e. things belonging to the domain of interest that can be promised independently. this points to a set of smallest independent entities that can make promises, and already gives some clues about necessary and sufficient distinctions.

   This principle can be applied to the computer systems that CFEngine's promises to configure, as well as to the elements in the language that models them, and the individual programs that form the software suite.

2. separate the different types of promises they can make.

   This allows us to define types of promise that can be grouped together for readability in the language representation.

3. identify the details of the promise body for each type. to do this, we reapply the entire method recursively; to identify the independent issues, or body-parts within the body of a single promise. we can do this because

each language representation of a promise is only a promise statement, and this statement must promise to be a clear and adequate representation of the promise it represents! self consistency is a key here.

4. re-group the independent agents/resources according to those that play simular roles and give the group or pattern a name for easy reference, i.e. identify which promises belong together to make the best use of patterns that reduce the amount of information used in the description.

5. use patterns to make generic promises that optimize the length of a promise expression.

This was the method used to generate the syntax for CFEngine's promise language, and it forms a method that can be re-used at all levels to bring about an atomization of a problem into necessary and sufficient sized parts.

13.3 A language of configuration promises

Suppose it were decided that a file should have a particular owner. in a declarative specification of this property, one might say something like:

```
files:

 mycomputer::

  "/directory/file"

     owner => ``fred'';
```

This assertion says that, in the context of the computer 'mycomputer', the named file in the named location should have the attribute owner with value 'fred'. this is a specification of desired state, and it is in fact what we would now call a *promise* about the desired state of the computer.

Here is a specific example from CFEngine itself. the following is a promise to create a file if it does not already exist, and that the permissions of that file will have certain specified values.

```
files:

  "/home/mark/tmp/test_plain"
     perms => mog("644", "root", "wheel"),
     create => "true";
```

the language is further generalizable. suppose CFEngine branched out into nano-technology and could build molecules, we might write a configuration to put together something like a water molecule.

```
molecules:

  "water"

     atoms => { hydrogen, hydrogen, oxygen };
```

The language can be understood as a way of abstracting away the details of *how* promises might be kept, i.e. what specific actions that need to be taken in a given situation, from a pure description of *what* promise needs to be kept. in this way, CFEngine reduces a context specific set of processes into a simple description of the desired end state. this is a highly compact form of description that places the focus on intended outcome rather than unknown starting point.

CFEngine's promise language has a generic and extensible form, as well as a fixed grammar, thanks to the use of the promise model. although it could easily be adapted to make any kind of promise, CFEngine uses it to describe the configurations of computers. the remainder of this chapter explains how promises are used in CFEngine.

13.4 The software agents

CFEngine consists of a number of software agents that contribute to the cooperative system for keeping promises. the role of these agents in keeping promises is explained in the subsequent sections. the main agents are as follows:

- `cf-agent` is the active agent in CFEngine which promises to read a collection of promise proposals (see section 10.6) about computer configuration for an entire network of computers, organized into named bundles. it further promises to pick out those promises that resources (promisers) on the same computer need to keep, and furthermore attempt to keep those promises on behalf of the respective promisers described in the proposals.

- `cf-serverd` is the server 'daemon' or service component[41], which promises to read bundles of promise proposals to grant access to file resources on a given computer. the server daemon further promises to mediate access to those file resources to authorized computers that need to download them in the course of keeping a 'files' promise.

- `cf-monitord`

  is the monitor daemon which promises to record and learn the patterns of resource usage on a computer, and then classify the current observed state in relation to the average learned state to inform about anomalous behaviour.

- `cf-execd`

  is the exec' (executor) daemon, which promises to execute `cf-agent` promise-keeping activities according to a predefined schedule.

- `cf-know` is the knowledge agent, which promises to read bundles of knowledge-related promise statements and compile these into a database, structured as a topic map (a form of electronic index). this includes a model of the global promise landscape for all the computers that share the same set of promise proposals.

13.5 The syntax of promises in CFEngine

Promises were used to redesign the syntax of the description language for computers' desired state. the syntax of CFEngine'd representation of promise statements grew from a pragmatic mixture of the pre-existing declarative language used by CFEngine before 2004 and the theory developed and described in this book. today, the CFEngine description language is a more or less clean implementation of the μ-promise theory described in this book.

Every promise has a promiser, and a body, which contains a type that refers to the basic configurable objects of the system (files, processes, storage, software packages, etc.). the type is a convenient header to group together promises of the same type under. thus a generic promise, in CFEngine, looks like this:

```
promise-type:

     scope::

      promiser -> { promisee(s) },

        # body or promise follows in remaining lines

        property => value,
        property => value,
        ...
        property => value;
```

Some examples help to see the domain-specific attributes and their allowed values in action:

```
files:

  "/tmp/test_plain"                    # promiser
                                       # (implicit promisee)
     perms => mo("644","root"),        # body
     create => "true";                 # body

processes:

  "snmpd" -> { "person","entity" },  # promiser -> promisees

     signals => { "term","kill" };   # body

packages:

  "apache2"                            # promiser

     package_policy => "add",          # body
     package_method => generic;        # body
```

While the precise details of these promise bodies are unimportant to readers who don't know CFEngine, the principles behind them are easily recognizable. the μ-promise model renders all of these diverse matters into a simple pattern so that everything about a system's desired state can be written in this form.

In the example above, the promise types are `files`, `processes` and `packages`. this type is a short explanation of the main intention of the promise, to be kept by the promiser, and detailed by attributes or property assignments in the body (containing =>) implicitly say what the type of promise is.

defining a variable is also a promise:

```
vars:
 "myvar " string => "string me along";
 "userset" slist => { "user1", "user2", "user3" };
```

indeed, everything represented in the CFEngine language is to be thought of as a promise.

13.6 How CFEngine interprets and keeps promises

CFEngine keeps two kinds of promises:

- promises that are hard-coded into its making and cannot be altered. these promises define the behaviour of the software itself, its defaults and decisions.
- promises that are read in as proposals from a trusted source and kept if they are deemed in scope.

One of the hard-coded promises that CFEngine keeps is to look for a file of promise proposals in a trusted location. the CFEngine agent reads in this file, if found, which may contain a potentially detailed model of the desired state for one or more computers. the promises described in this description file are only *promise proposals*, in the same way that promises in a contract are only proposals until they have been agreed to. the acts of reading this file constitutes a promise to agree to the proposals (by virtue of the hard-coded promise). the agent cannot be forced to read a file by any outside agent however, so this is an act of voluntary cooperation on the part of the agent.

The set of all promise proposals is usually called a policy, and the state of these promises being kept is called compliance with the policy. all computers can share a single policy, and the orchestration[42] of different roles within it is handled in the same way as a musical score – by labelling certain promises with certain players at certain times. in this case, the parts are configuration promises rather than musical promises and the players are instances of `cf-agent` running on different machines.

To handle the orchestration of parts, each promise is prefixed with a context expression or description of scope that describes which of the agents that should keep the promises. for example, if a promise is intended for all windows machines on monday's, one might write:

```
files:

 windows.monday::

   "c:\mydirectory"
      create => "true";
```

The line with double colons represents the scope for the promise to apply, or a description of which machines need to verify that this promise has been kept.

each agent evaluates these scope rules individually based on the context in which it finds itself.

Once a `cf-agent` extracts the promises it needs to keep from the total policy, it begins to process them one by one. the agent does not necessarily do this in the order that the promises are described, as the internal algorithms for keeping promises contain some embedded knowledge about how best to be able to keep promises.

Because of the principle that promisers can only make promises about their own behaviour, the promiser must always be the object that would be affected by a change. thus implementing a promise involves checking the current state of the promiser object (a file, process, package, etc) and comparing it to the promised state by working through the details of the promise body. if the current and desired state match, then no action needs to be taken and the agent simply reports that the promise was kept. if there is a discrepancy, however, the agent may either repair the state of the system or merely warn about the lack of compliance[43].

1. the agent expands any variables and assembles all the parts of the promise body into a single constraint list for verification against the current state of the promiser. the constraint list is a possibly long set of pairs of the form:

   ```
   attribute => value
   ```

2. it checks whether an object matching the promiser can be found.

3. the promised object may or may not be supposed to exist. if the promise says that it should exist, then it must be created. if it should not exist, it must be destroyed. both creation and deletion operations are 'idempotent', i.e. repeating them does not create or destroy things more than once, so the end result will always be that the object ends up existing or not existing, in accord with the promise.

4. the promiser generally has many complicated properties and attributes whose detailed state is part of the promise description. the agent thus proceeds to examine the current state of the promiser and compare it to the promised state. for each attribute, the question 'is this part of the total promise kept' can be answered by the comparison. again, there is a choice of whether to simply warn about discrepancies or whether to repair them. this decision too is part of the promise body, e.g.

```
files:

 windows.monday::

   "c:\mydirectory"
      create => "true",
      action => warn_only;
```

The default behaviour is to repair any non-kept parts of the promise body.

the precise details of what a CFEngine agent does in order to repair the state of a system go far beyond the scope of this book, into the realms of computer engineering.

A simple abstraction that can be made to keep promises is the notion of an operation that is 'convergent', i.e. that when repeatedly and unintelligently issued like a mantra will always result in the desired state[Bur04b]. this idea predated the notion of promises, but aligns well with the promise model. we can imagine a promise-keeping operation $\hat{o}_b$ for promise-body b which when applied to a state $q_{\rm any}$ results in the promised state $q_{\rm promised}$, i.e.

$$\hat{o}_b\, q_{\rm any} = q_{\rm promised}. \tag{13.1}$$

of course, the promised state also belongs to the 'any' state category, so we also have:

$$\hat{o}_b\, q_{\rm promised} = q_{\rm promised}. \tag{13.2}$$

These two relations constitute what is meant by convergence, and so we can interpret $\hat{o}_b$ as a promise-keeping operator. CFEngine's internal algorithms are based on the strategy of introducing one such operator for each independent promisable attribute of a promiser. creating these operators clearly requires some domain dependent knowledge in each case, but since the number of different cases is enumerable this is not difficult.

13.6.1 Who actually keeps the promise?

In the foregoing explanation, we chose a pure, promise theory viewpoint in which every promiser was considered responsible for keeping its own state promises. this is somewhat fictitious however. CFEngine deals with objects such as files, processes, disks, interfaces, etc, each of which will be represented by an autonomous agent in the sense of promise theory. these objects have no

capabilities to play an active role in maintaining their own state. what we can show is that this is just a formality, and the modelling is an appropriate fiction, as long as there is a simple trust relationship to the cf-agent program, which has the power to force a local system into compliance.

In CFEngine parlance, promises are actually kept by operators within `cf-agent`. an operator is a software algorithm that makes a change. these algorithms are located within `cf-agent`, not within the promisers (files, processes, etc), as the promisers do not have the capabilities to be self-repaired. however, this is no problem as the agents can always borrow this capability as a service by the powerful external agent. an operator then is easily explained as a bilateral service agreement between promiser and the powerful external agent: the promiser promises to inform the agent of its current value and accept any modifications from the agent. this is a simple indirection that is of mainly formal interest.

13.6.2 How long do promises last?

Promises in CFEngine last as long as they remain in scope. a promise can go out of scope in a number of ways.

- a description of scope, containing a time-dependent expression expires: e.g. consider the promise:

```
reports:

  monday.day22.march::

    "today is monday";
```

  this is a promise for CFEngine to report the string “today is monday” only on mondays when the date happens to be the 22nd of march. if this promise is kept, it would likely be a long time before it would be kept again, by which time the following might have happened.

- a policy gets overwritten by a new set of promise proposals that no longer contain the same promise, at such a time as when the agent voluntarily updates its policy. e.g. the following is a promise to update policy from an external source:

```
files:
```

```
any::
  "$(sys.workdir)/promises.cf"
     copy_from => cp("/source/promises.cf","remote_host");
```

an agent can update its policy when a local owner of the policy decides to change it, or when the local owner chooses to allow the agent to get updated instructions from an outside trusted location.

13.6.3 Coordination

By taking autonomy to its logical conclusion, we make very operation and every atom independent. but some more complex processes require coordination between promises. CFEngine handles this by using a dynamic scope as a method of communication between the agents. as the circumstances and status of machines changes, the scope of certain promises widens or narrows to alter the promises that apply to different agents[44].

The term classes is to some extent a misnomer in this context, born of the fact that there is only a single mechanism for cachine knowledge about the world in CFEngine. ignoring, the name, one may think of classes as a kind of flag mechanism that sets the context for a new promise proposal to be kept.

Here is an example of 'classes'.

```
bundle agent CFEngine_processes
{
vars:

  "components" slist => { "cf-execd",
     "cf-monitord", "cf-serverd", "cf-hub" };

processes:

  "$(components)"

      comment => "make sure server parts of CFEngine are running",
      restart_class => canonify("start_$(component)");

commands:

   "$(sys.workdir)/bin/$(component)"

         comment => "make sure server parts of CFEngine are running",
       ifvarclass => canonify("start_$(components)");
```

```
}
```

13.6.4 Promise bundles

In CFEngine, promises may be arranged into bundles. These bundles can be parameterized in order to make reusable components. This is a full implementation of the principles for componentization, as described in section 11.1.7.

```
bundle agent mybundle(parameter1, parameter2)
{

}
```

Similarly, promise bodies can be parameterized according to the two strategies for promise variation as described in section 11.1.7. A non-parameterized, or pre-packaged body would be written like this:

```
bundle agent mybundle
{
files:

   "/tmp/file"

     perms => safe_settings;
}

body perms safe_settings
{
mode => "0600";
}
```

This does not expose the choice of settings to the user in the promise itself. The alternative formulation, where the body component is parameterized would look like this:

```
bundle agent mybundle
{
files:

   "/tmp/file"

     perms => safe_settings("0600");
}

body perms safe_settings(parameter)
```

```
{
mode => "$(parameter)";
}
```

13.7 CFEngine's design components

One of the practical difficulties of using promise theory to describe large scale infrastructure as CFEngine does, lies in coping with patterns or repeated instances of whole bundles of promises. Even the most complex infrastructure has common themes that can and must be reused to compress the total information in the configuration, and this is to our economic advantage. We should not have to reinvent every wheel, every time. Consistency and compression of design is achieved by the parsimonious use of themes in an environment. Moreover, there might be multiple players in the system making equivalent or at least interchangeable promises. How do we choose between them?

The design of CFEngine tries to put a pragmatic face on this, and to use promise theory principles to unravel a potential mess. The notions of 'techniques' and 'design sketches' have been introduced by different users in order to break down a total system infrastructure into re-usable components. This allows infrastructure to be packaged as a kind of commodity.

We shall not refer to the specific implementations of 'sketches' and 'techniques' here, but rather look at what principles guide the constriction of any set of components of CFEngine promises.

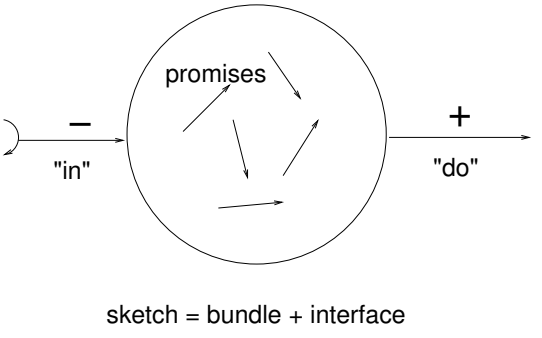


Fig. 13.1. CFEngine components are bundles of promises that cooperate to play a particular role in promising infrastructure. They use input from parameters provided by the environment and may other rely conditionally on other promises for keeping their own. The result is an aggregate promise to 'do' something, or represent a role.

Thanks to the basic sufficiency of promise algebra, CFEngine's sketches are little more than parameterized promise bundles, along with a number of guide-

lines for connecting them into an end-to-end system. By using a tool (the prototype is called `cf-sketch`) to combine sketches into a total design, adhering to the approved guidelines, one attempts to ensure the safe combinatorics of promise patterns. The basic micro-promise formalism makes makes this possible, but not necessarily trivial, as one must always deal with the issue of incomplete information. Naturally, promises can also be used to define these interfaces, so the entire problem is reducible entirely to one of keeping promises at different levels (see section 11.1).

13.7.1 Design guidelines for components

Based on the discussion in section 11.1, we can summarize some guidelines for the design of components in CFEngine.

Promises relating to correctness:

- Components promise a named with a unique root and a version number.
- Components separate data that will parameterize the components into a separate bundle that can be machine written for tool assistance.
- Components promise to advertise any exclusive promises made. Components should avoid promising to change globally shared resources in an absolute way. Shared resources should only promise minimal, relative state specifications. If that cannot be avoided, then the components should promise to make this information available to other components.
- Components that make conditional promises to use dependencies promise to refer to the specific, acceptable versions of the those dependencies.

Promises relating to usability cost:

- Components promise to have few parameters. Not every possible choice should be parameterized, else there is no benefit to componentization – choose rather more unparameterized components to cover diversity.
- Component names promise to have a relatively flat structure without a complex hierarchy.

As a meta-promise, the number of components should be kept as small as possible to avoid combinatoric complexity when composing them, but this should probably not be done by artificially collecting integrating many promises into large, monolithic, programmable components[45], as this is the opposite strategy to componentization.

13.7.2 An example format for components

We will abstract an idealized form for the components, without following techniques or design sketches directly. In fig. 13.2, we see an example pattern for a design component or 'sketch'. It has a number of features: a private namespace, and some well-known services or bunde names that represent how it promises to *use* data from outside (the − promise), and *give* service back (the + promise).

```
#
# Design sketch : Check named directories recursively
#

# Make names private to help avoid exclusive promises

body file control
{
namespace => "check_dir_changes";
}

#
# Input of data into clearly understandable variables from tools
#

bundle agent in
{
meta:

 "comment"      string => "Check a list of named directories recursively for change (also called a tripwire)";
 "version"      string => "1";
 "depends_on"    slist => { "CFEngine_stdlib" };
 "exclusive_promisers" => { cf_null };

vars:

 "directory_description[/etc]"      string =>  "System settings";
 "directory_description[/usr]"      string =>  "System base release";
 "directory_description[/bin]"      string =>  "System binaries";
 "directory_description[/sbin]"     string =>  "System operator special binaries";
}

###############################################################################################

bundle agent do
{
vars:

 "dir" slist => getindices("check_dir_changes:in.directory_description");

classes:

  "$(dir)_exists" expression => isdir("$(dir)");

files:

   "$(dir)" -> { "goal_infosec" }

       handle => "check_dir_changes",
      comment => "Change monitoring: $(check_dir_changes:in.directory_description[$(file)])",
      changes => default:detect_all_change,
 depth_search => default:recurse("inf"),
   ifvarclass => canonify("$(dir)_exists");

}
```

Fig. 13.2. An example sketch layout, with main bundles 'in' for receiving parameterized input and 'do' for detailing the promises.

13.7.3 Conditional promises and stacking of components

Sketches are containers, like promise bundles, and one is free to design them according to preference. It would be unwieldy for every sketch component to promise all aspects of every problem. Then there would be a lot of overlap between sketches. This is not necessarily a problem as long the promises do not conflict, but it makes sense to try to disentangle resources that can be considered shared into separate agents.

To keep to the promise principles then, the design of sketches ideally (but of course voluntarily) stick to the rule that any promiser that *can* keep its promise independently, should be represented by a different agent. In this case, that means that we would make separate bundles and sketches for separable issues.

Sketches might then still rely on one another (see section 11.1.9). For example, the promise of a database service might be offered by some agent as a direct promise to a user, or as a promise to an intermediary piece of software like a web service. Further, true to the notion that CFEngine is the combination of promises with patterns (grammar), we may try to keep the patterns as regular as possible.

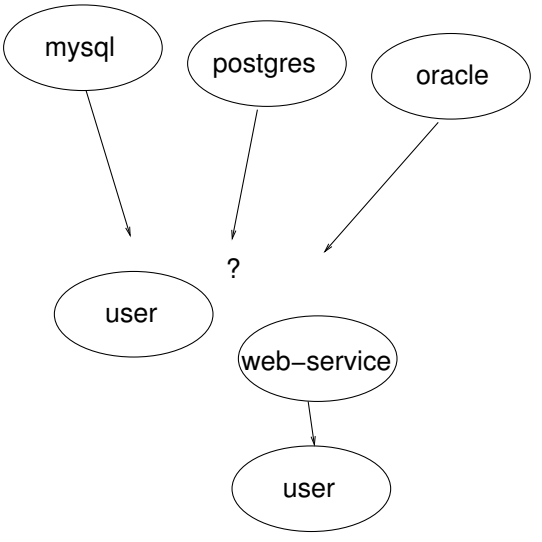


Fig. 13.3. If promise bundles play separable roles, then they should form different sketches. Similarly, several sketches might play equivalent roles as far as the promisee/consumer is concerned.

Formally, what we are doing is creating a palette of infrastructure patterns from *conditional promises*. Conditional promises say 'I will promise you b_1 as

long as someone keeps their promise of b_2 to me' (see section 6.1). Typical conditions for promising include:

- Needing data or guidance about the local environment.
- Needing component services forming part of a solution, so that multi-agent promises can be kept through cooperation.

What could have made cooperation potentially complicated in such a scheme is the lack of a simple model for composing patterns. This is where micro-promises' atomic construction comes to the rescue.

13.7.4 Composition of components

Readers are referred to section 3.13 for a discussion of promise conflicts, and section 11.1 on componentization.

There is some ambiguity in the way we define agents or components in a system. This arises from the ability for a collective og agents to represent a role. By cooperating, agents can make promises as if they were a single unit, e.g. an organization. In CFEngine, this means that we might draw the line between single design component and multiple cooperating components as we like. In electronics, one might have the analogy to whether one sells only transistors, resistors and capacitors, or whether one sells integrated amplifiers on a chip.

Design components represent functional issues that span a 'commons' of files and processes resources. The problem of shared system resources, comes into the fore in a major way as systems grow complicated. Some promising agents, like files, networks resources, databases, etc, promise services to multiple clients. This can lead to the possibility of conflict.

For example, a database might promise service as a standalone service, or it could be implicated as a back-end service to a web service (see fig. 13.3).

For example,

- You can't plug two different databases into a web server that only expects one, so a choice has to be made.
- A web service cannot believe a promise by two different intermediate agents to gain exclusive access to a shared resource at the same time, regardless of whether they know about each other.

These basic realities affect how sketches may be combined to provide cooperative promises.

It is tempting to think that any one of the services might do to quench the need for a database, but this is unlikely to be true. Since CFEngine promises are not unique to a single host, but probably span many hosts, searching for what ever happens to be out there would easily lead to *ad hoc* inconsistency, or an unknown desired state.

The key principles that illuminate most of the patterns in promise theory are that no agent (in this case no sketch) can promise something on someone else's behalf, and that promises are not guarantees and hence they might not be kept. Thus, when we intend to build cooperative systems, each part of the system has to assume the possibility that dependency promises will not be kept and make this clear to those 'downstream'.

A sketch, or bundle of promises, expecting input parameters from an outside agency thus has to validate things it depends on:

- That all the promises it relies upon from third parties are fully independent, i.e. that they are used to bring about a unique end-state, without conficts of usage.
- That, even if they don't directly conflict, no two dependency promises relied upon to quench a conditional promise b, make a promise they cannot keep by advertising exclusive access to a third agent's resource, e.g. two webservices that promise exclusive use of the same database. This would lead to contention for the resource that could not be detected directly by the agent promising b. It could however be inferred from accurate promises made by the dependencies.

These are the kinds of cooperation issues that are involved in trying to componentize a complex system. The situation is not so different from trying to design off the shelf components for anything: electronics, furniture, etc.

13.7.5 Accurate naming of sketch behaviour

The naming of sketch dependencies is equivalent to the accurate naming of a conditional promise made by the sketch. It needs to reflect its behaviour, and the behaviour it relies on from promises made by dependency sketches. To create proper expectations, a sketch should thus announce the conditionality of its promised behaviour to its promisees (see section 6.3).

We see that the accuracy of the expectations depends very much on the quality and specificity of the promises themselves. Sketches thus have to promise to advertise their services accurately in order to be successful at composition of services in an environment of limited scope.

It was tempting to try to form a taxonomy of sketches based on file names, however file hierarchies are mutually exclusive locations, which are poorly suited to promise behaviour[Bur12]. A hierarchy tries to put every sketch into a unique category that explains it. However, most categories are in the eye of the beholder and are at best ambiguous[46]. A flatter structure was suggested by both by CFEngine's history and by micro-promise principles.

CFEngine's existing approach fro this is to classifying properties, such as in classes promises that classify or tag the properties of environments. Each sketch promises certain meta-data 'tags' that announce and classify its intented purpose. This smoothly allows for multiple sketches to co-exist providing similar or even identifical promises.

In a promise view of the world, agents cannot be forbidden from making the same promises, it is up to the consumer of such promises to ensure that it has made a good choice.

13.7.6 The design sketch interface

...

13.8 Link to knowledge management

One of the key reasons to use promises as a model for the maintenance of a policy is to be a representation of knowledge. documenting intentions is a good practice in any sphere of management, but CFEngine's goal was to go further and use the model of interacting promises to form a description of the semantics of computers in a network.

In the regard, topic maps proved to be an inspiration[Pep09]. topic maps are a form of knowledge index representation, in which topics (or subjects) that can be discussed in some kind of document, are declared along with associations between them. such associations generalize the 'see also' linkage one sees in traditional indices.

Topic maps share a simple principle with the model of promises in that they atomize knowledge into non-overlapping issues, and then explain the relationships between them. if CFEngine is a form of chemistry with computer resources, the topic maps form a chemistry of 'documentable things'.

The main difficulty in using topic maps to represent knowledge, is the difficulty of classifying documentable things into a model that users can easily comprehend and employ. the complexity of most taxonomic decompositions of 'things' attains gigantic proportions with even a few different categories; thus,

expecting users to put things in the right boxes is expecting a lot. this is where the structure of promises comes to the rescue in two ways. promises offer a simple interpretation of the model of topics as well as a number of domain specific rules for constructing associative links between 'documentable things'.

13.9 Distributed deontic logic and non-locality

CFEngine was not alone in trying to solve the distributed management problem. simultaneously, and in parallel, the field of policy based management came up with the idea of user deontic logic to describe system configurations.

Deontic logic is a form of modal logic in which one describes things that must be, i.e. obligations, permissions etc. CFEngine rejected the notion of obligations early on[47].

In an obligation view, one would say, from the view point of an external lawgiver:

```
yourcomputer::

   /directory/file  must have owner => fred
```

This could seem like a small difference, but in fact it makes a huge difference to the consistency and implementability who makes the decisions about desired state.

Deontic logic (obligation) has two major problems:

1. obligations cannot be enforced without the permission of the owner.
2. obligations can lead to distributed conflicts of intent.

see the discussion in chapters 2 and 5.

In order to be able to cope with many diverse requirements, responsibility had to be decentralized. any central form of management would be a bottleneck in maintaining a desired state over time, because the states of computers do not remain constant or 'law-abiding' for long: programs crash, users interfere with settings, and a variety of other things occur that spoil the initial perfection of a computer configuration. CFEngine's task is to ensure that a computer are configured according to a desired state always – in other words, they would keep their promises.

The problem with obligations is that they may lead to distributed inconsistency, without ever nay agent in the system realizing it. it is necessary to have a complete and global view of the system in order to create a policy for more than one agent.

Promises solve this problem by allowing a set of proposals to be made either locally or globally, but which are always kept locally, and therefore are always visible in the same scope or context where they can be compared for consistency.

13.10 Compliance with requirements

We do not escape the need for obligation just by using a promise model. governments require and even legislate on the permissable state of computer systems that are involved in financial dealings and medical matters, for instance. there are minimum requirements for the security configurations of computers dealing with such matters.

These standards effectively become obligations, not in the computer but in the human realm of management. ironically, turning them into automated obligation logic is of little help because the kind of centralized power to enforce actually requires systems to be opened up in ways that actually compromise that security. we can, however, easily turn these requirements into promises.

Every promise about the state of the system can be categorized as being in one of three states:

- kept,
- repaired,
- not kept.

This leads to a simple model of compliance that has proved entirely satisfactory to auditors, who normally have no formal model to assure them at all. a major advantage of the promise model, here, is in connecting *assessment* of the system directly to expectations about the properties.

13.11 What does promise theory contribute?

Originally, the development of the theory of promises in this book was motivated by the desire for an algebraic approach to describing computer configurations. The hope was to find some kind of calculus for finding correct setups for computer systems that avoided the problems of previous approaches using deontic logic.

One of the advantages of promises over deonitic logic is in turning a theory of distributed constraints (deontic logic) into a purely local theory. This brings the ability to detect conflicts from syntax alone, with all relevant information localized in one place – at the point where the promise needs to be kept. Thus,

because the principle of autonomy says that no one can make promises for anyone but themselves, it can be applied strongly to any object in the system. Every promise is made directly by the object to which the promise applies. Thus the promiser is always the affected object in every declaration.

If a given object makes for than one promise of the same type, there is a chance that they could be in conflict. On the other hand, it is natural and even desirable that objects might make different promises in different locations or at different times. This is the role of the context to determine. Thus no two promsies of the same type and substance can exist in the same context without leading to an inconsistent state. This is a very simple rule that is easily policed. Since promises are an entirely local theory, all of the information is localized and there cannot be any distributed surprises.

A pleasant side-effect of this locality is that it brings the specification of intent closer to the stakeholders, who are typically the users or owners of the computers concerned.

Outstanding:

- The methodology of promises
- Chemistry developed enormously with the discovery of the periodic table
- The necessary and sufficient building blocks for constructing all possible outcomes.
- (Atoms) + chemistry
- Using atoms brings a maximum resolution of intent – we can talk about the smallest items that can keep promises, and therefore we have a bottom-up attention to detail that addresses necessary specialization.
- Atoms can be overridden locally, without loss of information

Notes

[41] A background program is called a daemon in unix parlance.

[42] The term orchestration has become common in describing distributed cooperation between computer agents. the term is inspired by the meaning from music, in which the players in an orchestra are able to play together by cooperating voluntarily. this cooperation is possible because each agent has a copy of the musical score (which may be considered a set of promise proposals for the agent to play certain notes, labelled with the scope of the instruments that should play them, e.g. first violins), and by the promise to use and heed hand-signals from a conductor, who is a coordinating agent. the main role of a conductor is to promise to 'wave in' certain players to remind them about when to begin playing, as well as to provide them with signals about loudness

in order to balance the lead. in CFEngine, orchestration is possible because each agent has a copy of the set of promise proposals and knows which promises are intended for it, i.e. which are in scope. a single point of coordination may also be used when it is important for agents to synchronize their behaviour in time.

[43]Compliance can never be guaranteed, because there are always issues beyond the control of an agent. a power failure would incapacitate an agent and prevent its promise of cleaning up a disk of junk files, for example. thus agents comply only on a best-effort basis. under normal circumstances, however, special algorithms in CFEngine have been designed to maximize the likelihood that promises will be able to be kept. this is called 'convergent behaviour' and is somewhat beyond the scope of the present book[Bur04b].

[44]CFEngine refers to scope or context expressions as 'classes'. we have tried to avoid the use of this term in this book, as the term is overloaded with connotations from a variety of sources.

[45]A VLSI approach to system configuration has no value in a component framework. In effect, the value of components versus simply designing from atomic primitives are mutually exclusive. Some users will choose to write a complete integrated set of promises for their organization, without breaking it into components, but then they would be unlikely to use any components made by others. Component users are typically users wanting to get ready made solutions. If you can make your own, you don't need them.

[46]For example, what would be the correct classification of the secure shell 'ssh'? Does it belong under security, shells, networking, communications, or some other category? In fact it could promise to be a member of any or all of them, and this is indeed what a promise approach would encourage.

[47]Growing up in a university environment, and trying to find a practical solution to researchers need to have the freedom to explore their own worlds, with their own resource, led CFEngine to reject the notion of centralized control early on. the notion of an obligation is something that comes from outside an agent and is something that the agent must obey, without having a voice in the matter. this kind of external attempt to force compliance leads to much conflict in organizations where workers need freedom to innovate.

14
Economics Aspects of Promises

In economics, one has two points of view: the Theory of the State in which a central power decides and rules by the use of force, and the Theory of the Firm in which behaviour is governed by agreements between peers[Ost90]. These two viewpoints mirror closely our distinction between obligation and promise.

14.1 Agent motivation

From the assumption of autonomy, we say that promises cannot be forced onto anyone: they are made voluntarily. What then is it that makes autonomous agents form promises, keep them or betray them?

Consider a system of agents behaving autonomously, in the sense of the foregoing chapters. We will show that promise graphs can be mapped onto economic games and hence rational economic principles can used as a tool for predicting the equilibrium outcome of general policies in agent interactions. Rational game theory is not the be-all and end-all of economic interaction, but it is a stepping stone that will allow us to address more complex decision models in future work, such as reputation and trust, as well as completely irrational choices.

One can map a valuation of promises onto a utility function or payoff-matrix for N-agent interactions, and hence find a relationship between games and promises. It would then be conventional to posit that the solution of such a game is a natural prediction of a stable policy, though it is known that there are many subtleties in formulating rational decision making problems. We summarize some results that are known from the theory of cooperation, based on two person games and suggest how these results can be interpreted in the framework of computer policy.

14.2 Promises with positive and negative value

A promise *valuation* $v_i\left(a_j \xrightarrow{b} a_k\right)$ is a subjective interpretation by agent a_i (in any local currency) of the promise in the parentheses. Usually an agent can only evaluate promises in which it is involved, as promises are only received (observable) by their recipient and their giver.

Definition 81 (Threat) *A threat is a conditional promise that an agent believes could have negative value to the receiver if true.*

14.2.1 The economic motivation for cooperation

Suppose the goal of an agent (such as a business) is to deliver a service S in exchange for some form of remuneration. The exact nature of the service is unimportant, e.g. delivery of a physical product, provision of information, etc); similarly, the exact nature of remuneration could be money transfer, exchange of goods, or even profit of something more abstract like "goodwill". The important feature is a trade of value that creates a binding[48].

The simplest way to model this kind of interaction is view it as two promises, back to back: a promise to provide a service and a promise to pay. In fact, most parties will only make promises on the condition that they are promised something in return (see fig 14.1), so we can add the notion of conditionals also. This will have considerable consequences for the complexity of the agreements between the parts of the business.

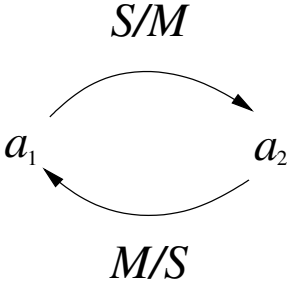


Fig. 14.1. A simple business relationship with a client: the business provides a service if money is promised, the customer promises money if service is promised.

In the basic configuration, A promises B the service S if it receives money M, and B promises A money M if it receives service S. This of course results in deadlock which we shall return to at the end of this chapter. Assuming that

this deadlock is broken, i.e. that one of the agents begins by keeping its promise, this loop becomes a valuable economic cooperation.

14.3 Voluntary coooperation and Managing Services

Consider an assembly of agents, denoted n_i, $i = 1, \ldots, N$, each of which is entirely responsible for its own behavioural policy towards neighbouring agents (i.e. exhibits complete autonomy). No agent takes orders from any other. Examples of such autonomy can be found in software like cfengine[Bur95, Bur93] and the Border Gateway Protocol[Hui00].

Agents exhibit cooperative behaviour by making promises to a neighbour, if they perceive a value in doing so to themselves. Hence each agent must be capable of evaluating the worth of promises and services made to it by other agents, in relation to its own load of commitments. We assume that each agent has its own algorithm for making this judgement. Let us ask:

1. What mechanisms are needed in policy for the agents to *start* cooperating (emergence)?
2. What is necessary for the agents to *keep* cooperating, once started (stability)?

These are typically macro-level economic concerns. They concern what is *profitable* for a group as a whole. Yet there is no implication of governance of the assembly of agents from a central point, or commiting them to a plan of authority that is decided externally. Each individual agent is concerned with only:

1. What *benefits* can I gain from interacting with my peers?
2. What might I *lose* from interacting with my peers?
3. How should I *behave* to maximise my gain, or minimise my loss?

Service provision is about committing to behaviour that is valuable to a client, but it can only be sustained if the provider also receives some benefit. We use promise theory to define interactions between a number of agents that are initially peers, with no predecided roles.

14.4 Other approaches to cooperation

In the computer science literature, several attempts have been made to encourage or enforce cooperation in so-called 'autonomous agent' systems. However, the

meaning of autonomy is often less strict than ours, e.g. implying only that agents work independently, but can still be commanded. A common approach has been to assume certain functionality in the hardware to make it necessary for the nodes to cooperate in order to have the desired functionality. One example is the so-called *nuglet counter* which increases for each time the node forwards a packet on behalf of another node, and decreases when the node sends one of its own packets[LST]. Unless the nuglet counter has a value greater than zero, the node is not able to use the network for sending its *own* packets. This is an example of enforcing cooperation between nodes, but this is in no way an example of *voluntary* cooperation. It relies on having some sort of homogeneous tamper-proof hardware for all nodes, which makes it impossible for each node to not cooperate. It implies an external authority which is obeyed implicitly by all of the agents.

Another approach which is closer in spirit to voluntary cooperation, is the so-called *reputation-based* system. Such systems implement mechanisms for discovering and logging bad behaviour in the agents, which again can be used as aid for other nodes to punish bad behaviour. Peer-to-peer (P2P) content distribution systems often work in this way. The main purpose of each system is to distribute some piece of content to as many peers as possible during some period of time. The strength of these systems is their efficiency (it is a form of epidemic spreading), a result of the fact that all peers with a downloaded piece keep uploading to other peers, and hence contribute to further distribution. One important reputation factor, is how long the peers who have finished downloading stay connected after they are finished. There are several different approaches to keep the nodes connected for as long as possible[LST].

14.5 The value of a promise

The key point that makes economics a factor is that promised services are valuable commodities to individuals. How shall one measure the value of a promise to the giver or the receiver? In promise theory, a promise documents a *constraint* on a set of values; it has no a priori *value* in any form of recognized currency, as such a value would be a subjective matter. Indeed, autonomous agents have *a priori* no common concept of universal currency. Each agent is free to decide on some valuation of a promise as it sees fit. It could be a measured level of service quality, or any function of the service quality. A promise by a ‘famous’ or ‘royal’ agent could be worth more than a promise by a ‘commoner’. Alternatively it could be a heuristic such as an arbitrary scale of ‘business relationship value’ or other subjective valuation. Such subjective valuations are the basis of

economic markets, so we wish to allow these kinds of heuristic judgements.

In ref. [BFa], we alluded to a strategy of simplifying the detailed *typed* view of promises by replacing it with a common currency graph, in which each agents's specific promises are collapsed into an overall view of its essential reliability. The numerical valuations translate into weights on the edges of the graph. There are different ways of determining the edges on these graphs:

- *Estimated edge values*: We find a probability that pairs of agents will keep promises of various types. This could be based on observation, unsupervised learning, or even simulation. However, this requires a knowledge of some kind of history of actual agents.

- *Negotiated edge values*: Another possiblity, which captures the essence of a system with less initial input, is to consider a worst case scenario in the form of a bargaining game.

To study this further, assume that each agent n_i possesses a *valuation function* $v_i(\cdot)$ which it can apply to promises given and received in order to determine their value. In general, an agent can only evalute this function on a promise that it knows about, i.e. one that it makes $v_i(n_i \xrightarrow{\pi} n_j)$ or one that it receives $v_i(n_j \xrightarrow{\pi} n_i)$. The currency of this value v_i is private to the agent n_i concerned.

In some cases, agents will have to agree on a currency standard in order to make agreements. Trading of promises will turn out to be important to stability and reliability—i.e. whether agents choose to keep their promises.

Example 43 *The value to Internet Service Providers in Border Gateway Protocol peering promises is known to be based more on fostering good relations and business acumen than on technical measurements[Nor01a, Nor01b]. It is important to account for these business intangibles in modelling economic collectives.*

We believe that one should take seriously a multi-currency view of promise interactions.

14.6 Contract Economics

There is a lot of economic theory about contracts, but it seems relatively new and no one is completely certain about it... still, some interesting things here.

The models seem to lack the symmetry of promises, and so promises have an advantage over them for generality. The talk about commitments and "effort" however..

Definition 82 (Trade) *A trade agreement is a bilateral promise to exchange goods of value. Trade is instantaneous. Goods are money.*

A principle of trade is that traded promise bundles can be decoupled using money. i.e. if we trade gold for textiles, one can decouple these movements of goods simply by paying their equivalent value.

Definition 83 (Peering agreement) *A bilateral commitment between two parties (peers) to serve one another with possibly different services without the exchange of money.*

Definition 84 (Paid service agreement) *Agreements are usually a commitment to serve, and a promise to pay.*

Definition 85 (SERVICE AGREEMENT) *is a binding agreement between a service provider and a client*

Definition 86 (SERVICE LEVEL AGREEMENT) *is a subset of SERVICE AGREEMENT which concerns the scope of the commitment made by the service provider and the scope of the promise to use the service by the client.*

14.6.1 Principal-agent model as promises

A model that gets mentioned a lot is this one. It is about one "agent" controlling another.

Principal is someone who wants to employ an agent to perform a service over which he does not fully have control. The principal (manufacturer) hires the agent to sell its product for instance.

For instance, a distribution agent might advertise less than they are conracted to do in order to *free-ride* off the manufacturer's reputation. These principal-agent problems are often dealt with by "vertical restrictions" such as forbidding agents to change the price of goods. Courts have been uneasy about these kinds of contracts since they seem to oppose competition.

Forbidding this has little effect, since comapnies can simplt integrate vertically and handle their own distribution.

The profit maximization hypothesis says that all firms/agents should try to maximize profit. If they don't it is explained only by bad management.

14.6.2 Incomplete contracts - guidelines

Partial coverage of promises – guidelines. People are starting to realize there are things that are beyond their control. However, the models believe that this is because companies think it is too expensive to control everything...

14.7 Trading

There are many possible valuations of promises – the extent to which promises are kept, how much they are worth to each party etc.

14.7.1 Details needed to keep promises

For example, suppose n_1 gives n_2 some money. What is it for? Has n_1 kept a promise to n_2? We clearly need some more detailed notions to understand this points.

We need more than cash flows to model economics – we need to be able to trace the 'types' of or reasons for transactions. We need to know when promises are kept. This explains the need for accounting.

Any financial system must be sufficiently sophisticated to allow the notion of a promise being kept.

Example 44 (Financial payments) *Suppose n_1 wants to make payments to n_2 for a certain reason r, by a certain deadline d, of a certain amount a. We may a promise type and constraint $\pi = \langle \tau(r,d), \chi(a) \rangle \leftrightarrow \langle r, d, a \rangle$. A type can be a parameterized compound type, like a table in a database.*

What if over a certain history n_2 received a number of payments that add up to at least the amount a? Is this sufficient to be able to say that the promise has been kept? We need to label the transactions in relation to the promise.

The fact that the types are intrinsically important shows that one must always have a number of different type names, so it does not matter whether the currencies belonging to those types are the same or not. We could pay in dollars, euros, coffee, gold, silver...just as easily

What about negative promises? these usually come about as a result of *design rules*. We might promise never to pay for sex, never to pay a bribe to a member of parliament... something that is against the law or the design of the system.

Example 45 *The availabiliy of knowledge is important. Suppose I promise to never call my friend when he's are sleeping. This is not possible based on information available to the agent, unless there is some handshaking. i.e. the friend*

must promise to reveal the information about when he is sleeping, because it is a conditional promise, and I must promise to use that information made available to me.

This will lead us to the need for handshakes, protocols and bilateral agreements. Promises are most likely to be kept if they are reciprocal.

14.8 Money and work as promises

Money can be viewed as a promise, i.e. as a service that transfers information about transactions. It has the property of being *countable*. On the British pound note it says: "I promise to pay the bearer the sum of One Pound", indicating that the money tokens are not the source of actual value themselves. Although money is just another service with a particular role to play, it has a privileged position in modern economics, so it is of interest to distinguish it.

A promise to pay money (for some service) is handled in instantaneous transactions. It summarizes instantly something that took effort to produce and serve. However, one can also imagine a promise to pay off a debt by working over time. Here the promise is the work, not the act of paying.

Or we can commit to being able to pay the wages of our employees if it involves some effort. You do not need to expend effort to make a transfer. The commitment to pay an employee is the commitment to earn the money that you pay! The employer *promises* to pay them. An effort is required to perform the service. A capitalist shows no commitment to his employees. Does this explain the bureaucratic mentality? A false impression of doing work to imply a commitment!!

A definition of money might be simply 'a countable instantaneous transfer service'. The value of money is set in exactly the same way as any other service – by a valuation. Money and information are defined in the same way! But money has to have certain numerical properties and it has to be used in a certain way (role).

14.9 Trading promises and contracts

The concept of a contract was defined in section 10.6.1. A contract is a tool for both establishing cooperative goals between independent parties or agents, and for imposing penalties when goals are not met. They are a method of codifying voluntary cooperation between peers in a social network[Axe84] to form cooperative relationships and communities. Service Level Agreements are examples of contracts for service provision between IT peers.

Determination and optimization of contracts mixes both human and technological issues[ea, Rod02, DKS02]. In particular, the terms of an agreement can be quite sensitive to the human and technological context in which they are made. A great variety of automated technologies can be involved in automated contract exchange, motivating the need for a *lingua franca* for agreement. This is often achieved with *ontologies*[Str07].

Deontic logic, the logic of obligations and counter-obligations, has been used as a framework for making the human language of contracts precise, often in a legal context[DS97]. In ref. [SB04] contracts are used to define penalty costs for failing to live up to their obligations. This is a common theme in economic models of promise breaking[CD] (see below). A continuum study of this kind of contractual penalty feedback between a pair of peers has been examined in ref. [BBJF]. There the authors noted the potential instabilities in relationships between agents occurs when they were allowed to impose punishments and penalties on one another.

Community policy is often formed through agreements. It can be thought of as a kind of 'law making' either by mutual cooperation of by external lawgiver. The Law Governed Interaction project[MP00, Min05] considers, for example, how policy seeds a community through the conformance of its participant with a global standard. The external law giver can, on the other hand, be seen as a special case of concensus through voluntary cooperation. In promise theory[Bur05, BFa, BFb], which has qualitative similarities to 'commitments' in multi-agent theory[Fer99, Woo02], one assumes that every agent decides for itself how it should behave. Stable communities can then occur spontaneously if the conditions for voluntary cooperation are sufficiently conducive. Here one considers the stability and consistency of a series of agreements in a network, to see whether the sum of individual promises can lead to a consistent union of ideas.

Consider two promises, back to back as in fig. 14.2. This diagram represents

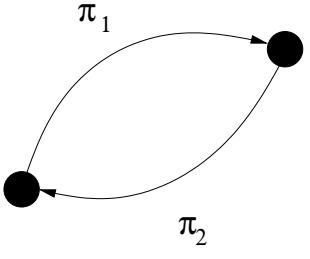


Fig. 14.2. A trade of promises behaves like a two-person bargaining game.

a peering agreement (e.g. such as one might expect in a Border Gateway Protocol (BGP) configuration between two service providers). Alternatively, it could represent promise of service for the promise of payment. Promise types should generally not overlap, so as to avoid the possibility of contradictory or broken promises.

Typed promises form disjointed graphs, but bargaining agreements clearly motivate the unification of promises using a kind of *common currency* or valuation of the promises. One therefore arrives at a natural unification between a service viewpoint and an underlying economics which can motivate the *reliability* or *duration* of promises over time[BFb]. In promise theory, a *commitment* is a promise that requires an irreversible investment of resources, i.e. it has a non-returnable cost associated with it.

14.9.1 Contract roles

A business that provides a service makes promises to various parties. These promises are usually made in the form of legally enforcable contracts, but the success or failure of such contracts has little to do with the fact that they are legally binding. Both the provider and the recipient of a service have an interest in maximizing their gain with regard to a business relationship and this is the primus motor for successful contracts.

What is interesting in such a situation is that it is not always a symmetrical exchange. One of the parties is usually in possession of an informational advantage over the other, and who makes their move first can completely change the optimal outcome for the parties. There is a risk involved in making a judgement.

14.9.2 Principle agent model

In economic game theory, the *Principal Agent Model* is a basic model for highlighting the issues surrounding voluntary cooperation between agents making promises to one another[Ras83, BG02]. It is a simple (indeed simplistic) model of a contract negotiation. The basic idea is that a *principal* (agent) wants to outsource a task to one or more (service) agents, and that the service agents fall into a variety of different types that indicate their relative costs and abilities.

Some agents, for instance, might be better qualified than others and so demand higher wages, while others might be able to do the same job more cheaply. An obvious case is the trend of outsourcing to Asia in which agent types represent the living standards cand costs of their respective countries. One has expensive Western agents, and cheap Eastern agents, both of whom are capable

of doing the same job. This leads to a decision dilemma for the principle. If the principle offers too little for the task and no cheap agents are available, the work will not be done. If the principal offers a higher rate, then even the cheap agents will pretend to be expensive agents in order to make a higher profit.

Thus, if the principal makes its promise of payment before the agent agrees to perform work, then the principal has a disadvantage. This, of course, is a normal state of affairs. So what can the principal do to try to coerce agents to provide their service for an optimal price?

Information is key in voluntary relationships because there are decisions to be made by each agent (as promise theory predicts). An Asian company could easily pose as a Western company in order to receive a better price for their services. Why would they not do this, given the chance? Economic game theory has addressed this question, in what has now become contract theory.

Consider a principal P and two agents A_1 and A_2 of different types θ_1 and θ_2, where θ_2 is a type of agent with a high price and high overheads, and θ_1 is a low price, low overhead supplier type.

There are two possible scenarios, depending on who starts the negotiation (see fig 14.3). In scenario (a), the principal first promises separate deals for the two agents without knowing their types, then the agents can accept or refuse. In scenario (b), the agents first reveal their true types and then the principal offers them a deal accordingly. The dotted lines denote conditional promise responses.

Fig. 14.3. The principal agent model's bilateral promises.

In (a) the principal is at a disadvantage, since the agents have not revealed any information about themselves. In (b), the agents are at a disadvantage since the principal can offer them the minimum rate at which it knows they will work (this is called the first-best solution for the principal). In most cases, a client (principal) is in the position of not knowing an agent's overheads, so (a) is the likely scenario. We can analyze (a) further by making a general promise scenario.

The potential promises made by the principal are about wages (money) m_1, m_2 to the agents, whereas the potential promises made by the agents are about "effort" s or the agreed Service Level for the principal. In a contract situation, both agents will make promises conditional on what the other delivers:

$$\begin{aligned} P &\xrightarrow{m_i/s_i} A_i \\ A_i &\xrightarrow{s_i/m_i} P. \end{aligned} \tag{14.1}$$

where $i = 1, 2$. Let now the value of a promise to an agent is denoted by v_a. Then the economic value of the these promises may be written:

$$\begin{aligned} v_P\left(P \xrightarrow{m_i/s_i} A_i\right) &\mapsto -m_i \qquad \text{if } s_i \\ v_A\left(P \xrightarrow{m_i/s_i} A_i\right) &\mapsto +m_i \qquad \text{if } s_i \\ v_P\left(A \xrightarrow{s_i/m_i} P\right) &\mapsto +s_i \qquad \text{if } m_i \\ v_A\left(A \xrightarrow{s_i/m_i} P\right) &\mapsto -C(s_i). \end{aligned} \tag{14.2}$$

Here $C(s_i)$ represents the cost of exerting the effort or service level s_i to the agent, i.e. this promise is in fact a commitment once made. The cost is normally represented by some convex polynomial function of s_i such that greater effort leads to a more than proportionally high cost (this is the opposite of economies of scale), i.e. it hurts to work hard, so the agent want to be paid much more for harder work than for light work. For definiteness, we define the simplest quadratic form for this exposition:

$$C_i(s_i) = \frac{1}{2}\theta_i s_i^2. \tag{14.3}$$

Notice that the cost is proportional to the type or "overhead" parameter, which might be real or fake.

The valuations of each agent's promises (given and received) now provide us with an optimization problem, which each autonomous agent valuates in its own terms. This is a so-called *adverse-selection* game whose outcome essentially decides whether the proposed promises will be made[BFb].

The total payoff for the principal is the sum of its promise valuations. It amounts to the difference between the effort received from the two agents and the wages paid to them:

$$\pi_{\text{principal}} = \sum_{\text{promises}} v_P = \sum_{i=1}^{2}(s_i - m_i). \tag{14.4}$$

The vector of values $\vec{m}$ with components m_i represents a *contract specification* in relation to the promises above[Ras83, BG02]. The payoff to the body of contracting agents is

$$\pi_{\text{agent}-i} = \sum_{\text{promises}} v_A = m_i - C(s_i), \tag{14.5}$$

i.e. the wages received minues the cost of expending the service level effort.

The problem for each autonomous agent, including the principal, is to maximize its own payoff in this exchange. Agents are normally thought of as being risk-averse. That is, they choose to avoid options that could reduce their payoff. Incentives can help the principal convince the agent of a low risk (when the agent does not mind bearing all of the risk of a transaction it is said that the problem is solved by "selling the firm", i.e. the principal no longer needs to be involved). The solutions of this simple model offer a basic insight into incentive management.

'First best' solution In the first best case, the agents announce their types and make clear that they will only accept a contract that will not give them a negative payoff:

$$m_i - C_i(s_i) \geq 0. \tag{14.6}$$

This is known as the *participation constraint*. However, since the principal knows the agent's overheads it can simply choose m_i to render this an equality, thus bleeding all of the agents' profit with a minimum wage. This kind of price discrimination is normally forbidden by law – a principal is not allowed to offer a different price for the same deal to different agents. It is however a possible solution to the problem.

Meanwhile the principal maximizes, subject to the agent's constraint for participating. The Lagrangian for this is

$$L = \sum_i (s_i - m_i) + \alpha_i \left(m_i - \frac{1}{2}\theta_i s_i^2 \right), \tag{14.7}$$

where α_i are Lagrange multipliers. Solving:

$$\begin{aligned} \frac{\partial L}{\partial m_i} &= -1 + \alpha_i = 0 \\ \frac{\partial L}{\partial s_i}\Big|_{e^*} &= 1 - \alpha_i \theta_i s_i = 0 \end{aligned} \tag{14.8}$$

i.e. $s_i^* = 1/\theta_i$. This gives the level of effort that the agent is bullied into providing, for minimum wage $m_i^* = 1/2\theta_i$.

This solution assumes that the agents have told the truth about their overheads, or that their true costs have somehow been determined by the principal. It is easy to see however that if the principal simply guesses a contract without knowing the real agent types, it can be tricked. A cheap agent can claim an expensive contract, giving it wages minus its cheaper costs at the lower effort level, giving it a non-zero profit:

$$\begin{aligned}\pi_1 &= m_2^* - \frac{1}{2}\theta_1 s_2^* \\ &= \frac{1}{2\theta_2}\left(1 - \frac{\theta_1}{\theta_2}\right) > 0. \end{aligned} \tag{14.9}$$

Ironically the contract m_i awards the higher wages to the cheaper agent, who also gets the bulk of the work if work is to be divided amongst several agents of the same type. This solution places the principal in the position of advantage over the contracters. Of course, the solution assumes that agents of the appropriate types exist and are available for carrying out this work. A principle might have to cover a task by employing several agents of different types in practice. In that case, they would be optimally assigned according to the proportions above.

Second best solution - adverse selection In the second best solution, the agents do not reveal their abilties or costs to the principal up front. It is up to the principal to prepare a contract to a general mass of agents who might be of different types. We can assume that the principal has done some market research so it knows the different types θ_i and it knows roughly the probabilities p_i which which they occur. We can now show that it is possible for the principal to come up with a contract that dissuades the cheaper agents from pretending to have high overheads so as to trick the principal into offering the higher wage. The strategy is to use the right incentives. Thus we must add an additional *incentive constraint* with the the principal, which says that an agent will obtain the maximum payoff by not pretending to be of a different type, i.e. the principal requires that

$$\begin{aligned} m_1 - C_1(s_1) &\geq m_2 - C_1(s_2) \\ m_2 - C_2(s_2) &\geq m_1 - C_2(s_1) \end{aligned} \tag{14.10}$$

This constraint can be satisfied. Now, in this scenario, there might be many agents bidding for work, and the principal knows only the probable distribution of these p_i, thus it can now compute its *expected payoff*:

$$\langle \pi_{\text{principal}} \rangle = \sum_i p_i(s_i - m_i), \tag{14.11}$$

subject to the contraints above and the assumed participation constraint from before $m_i \geq C_i(s_i)$.

Let us solve this for two types θ_1 and θ_2. To determine the contract offer (m_1, m_2), the principle has sufficient information from its market research to maximize eqn. (14.11) subject to the following constraints:

$$m_1 \geq \frac{1}{2}\theta_1 s_1^2 \tag{14.12}$$

$$m_1 - \frac{1}{2}\theta_1 s_1^2 \geq m_2 - \frac{1}{2}\theta_1 s_2^2 \tag{14.13}$$

$$m_2 \geq \frac{1}{2}\theta_2 s_2^2 \tag{14.14}$$

$$m_2 - \frac{1}{2}\theta_2 s_2^2 \geq m_1 - \frac{1}{2}\theta_2 s_1^1 \tag{14.15}$$

$$\theta_2 > \theta_1 \tag{14.16}$$

$$p_1 + p_2 = 1 \tag{14.17}$$

If we add (14.13) and (14.15), then it follows that $s_2 \geq s_1$, i.e. the expensive agents must work at least as hard as the cheaper agents in an optimal solution. Moreover, it is easy to see that not all of the constraints can be independent. Substituting eqn. (14.12) into (14.15), we get

$$m_2 \geq \frac{1}{2}\theta_2(s_2^2 - s_1^2) + \frac{1}{2}\theta_1 s_1^2. \tag{14.18}$$

Thus we may maximize the Lagrangian:

$$L = p_1(s_1 - m_1) + p_2(s_2 - m_2) + \alpha\left(m_1 - \frac{1}{2}\theta_1(s_1^2 - s_2^2) - m_2\right) + \beta\left(m_2 - \frac{1}{2}\theta_2 s_2^2\right) \tag{14.19}$$

The new equations are therefore:

$$\frac{\partial L}{\partial m_1} = -p_1 + \alpha = 0 \tag{14.20}$$

$$\frac{\partial L}{\partial m_2} = -p_2 - \alpha + \beta = 0 \tag{14.21}$$

$$\frac{\partial L}{\partial s_1} = p_1 - \alpha\theta_1 s_1 = 0 \tag{14.22}$$

$$\frac{\partial L}{\partial s_2} = p_2 - \alpha\theta_1 s_2 - \beta\theta_2 s_2 = 0 \tag{14.23}$$

Substituting eqn. (14.20) into (14.22) gives the optimum effort

$$s_1 = s_1^* = \frac{1}{\theta_1}. \tag{14.24}$$

and substituting eqn. (14.21) into (14.23) gives

$$s_2 = s_2^* = \frac{1}{\theta_2 + \frac{p_1}{p_2}(\theta_2 - \theta_1)} < \frac{1}{\theta_2}. \tag{14.25}$$

To find the contract rates m_i, we note that maximization of the Lagrangian implies minimization of m_i, so the constraint inequalities become "shrink wrapped" to equalities in eqn. (14.18) and (14.12), giving:

$$m_1 = \frac{1}{2}\theta_1 + \frac{1}{2}(\theta_2 - \theta_1)\left(\theta_2 + \frac{p_1}{p_2}(\theta_2 - \theta_1)\right)^{-2} \tag{14.26}$$

$$m_2 = \frac{1}{2}\theta_2\left(\theta_2 + \frac{p_1}{p_2}(\theta_2 - \theta_1)\right)^{-2} \tag{14.27}$$

$$\tag{14.28}$$

We note that $m_1 > m_2$, thus we pay the cheaper agents a surplus to prevent them from choosing the θ_2 deal. This amount is just enough to make a rational agent choose the deal designed for it.

Comments Although this principal agent model is simplistic, it makes the interesting point that optimizations are possible, even when incomplete information prevents a business from achieving a theoretical maximum. The model assumes however, from the underlying promise constraints, that wages will only be paid if the agent does the work it promise. The model is flawed in a number of ways. It assumes that work can be given in a continuum and that the required level of work can be assigned into bundles s_i that are suitable for the available agents. There are many reasons why this simple exchange is too simple. The cost function $C_i(s_i)$ is taken to be convex as an incentive, but in fact many operations will have an opposite experience: economies of scale will allow the agents to have concave costs.

Notes

[48]In physics the bindings are motivated by energy considerations. Energy is the economic currency of physics.

Notes

[48]In physics the bindings are motivated by energy considerations. Energy is the economic currency of physics.

15
Promises and Mathematical Games

This is based on [BFb].

15.1 The relationship between games and promises

The mechanism of rational, voluntary decision-making used in economics is the theory of games[Ras01, Mye91]. In this section, we would like to show that there is a natural relationship between game theory and certain constellations of promises which provides a motivation for keeping those particular promises out of all the promises an agent might make. We confine our discussion here to normal or strategic form games.

We alert the reader to a potential confusion of notation here. The symbol π is normally used for a promise. In the context of game theory it is often used also for the payoff or utility matrix. We shall therefore use the symbol $U_{ijk...}$ for the payoff or *utility* matrix of an N player game.

We follow Rasmussen in our description of games to avoid repeating well-known definitions[Ras01]. A game is a number of players n_i, where $i = 1 \ldots N$, a set of actions, payoffs and information. A player n_i's *action set* is the entire set of actions or pure strategies available to him. An action combination is a set of one action per player s_i in the game. Player n_i's payoff or utility is denoted $U_i(s_1, s_2, \ldots, s_N)$ and has one of two interpretations:

- It is the utility received at the end of a trial by a player, based on what the player did.

- It is an estimated or expected utility to be considered as a function of the possible or expected strategies a player might choose in a future game.

It is this second interpretation that most closely relates to promises, since we shall identify promises with expectations of actions. However, we should be clear that the latter cannot exist without some measure of the former. This duplicity of interpretation mirrors a similar one in promise theory, in which one can measure the reliability of a promise with a scalar probability value. The probability can then be regarded as a history of a player, or as a prediction of future behaviour.

A *strategy* is a rule that tells player n_i which action s_i to choose at each instant in the game, given his or her information set. An equilibrium $s^* = (s_1, s_2, \ldots, s_N)$ is a strategy combination consisting of the 'best' strategies for each player, with respect to the expected payoff. Several such equilibria might exist.

There is a class of games in which the possible actions or moves of a player are the same in each round of a game. Thus each round of an extended game is indistinguishable from multiple games, except perhaps in the information available available to the players. Iterated games are special because both *each round* and the entire *iterated game* can be represented as a utility matrix of the same form. Thus we may discuss both extended interactions and one-shot interactions using a utility matrix formulation as long as we restrict attention to such games. We shall do this here.

Promises are static snapshots of policy. A promise graph such as that in fig. 15.1 can be viewed in two ways: either as an equilibrium over a complete inter-

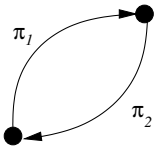


Fig. 15.1. A simple exchange of promises.

action epoch or as a single round in an repeated interaction. To make the move from games with actions to games with promises, we simply replace actions with promises. Any ordering of promises (who makes which promise first) is analogous to an ordering of actions. In a game one considers a strategy to be a choice of actions. Here we shall assume it to be a set of promises made:

$$n_i \xrightarrow{\{\pi\}_i} n_j \tag{15.1}$$

where $\{\pi\}$ is potentially a bundle of different promises made from one player to another. This bundle might also comprise the empty set.

The payoff or utility received by the ith player or agent in the system is the sum of *its own* private valuations of promises made and received:

$$U_i(\{\pi\}_1, \{\pi\}_2, \ldots, \{\pi\}_N) = \sum_{j \neq i} v_i \left(n_j \xrightarrow{\{\pi\}_j} n_i, n_i \xrightarrow{\{\pi\}_i} n_j \right) \quad (15.2)$$

Any kind of promise can be evaluated, regardless of whether it is a service promise or a use-promise[Bur05] (give or take promise).

15.2 Classification of games

Games are classified in various ways, particularly with regard to cooperation and conflict. An interaction is *cooperative* if players coordinate their actions in order to improve their final payoff (utility). Cooperation might be contingent on available information. An interaction is a *conflict* if the actions or intentions of one player can adversely affect the payoff (utility) of another player's actions. Let us give some examples of these representations of games in terms of promise exchanges.

15.2.1 Cooperative, no conflict

Consider two agents helping one another to provide a sufficient level of service for a third party. By helping one another, they are able to work together to meet 3's requirements and secure contracts with the third party and hence both profit: they are not in competition with one another. This scenario is shown in fig. 15.2.

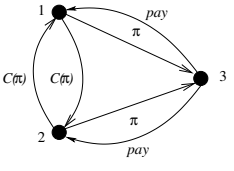


Fig. 15.2. Cooperative game, no conflict.

Two agents 1 and 2, make coordinated promises to serve node 3. The third party promises to reward both agents for their service, independently. The values of the remuneration promises $v_{1,2}(pay)$ are not related.

15.2.2 Cooperative, with conflict

Consider now two agents who bargain with one another directly for services from one another. Both need these services from the other, so the services are

valuable (fig 15.3). By making each other's promises contingent on what they receive from the other, the value each of them receives becomes moderated by the other player. If they cooperate with one another, they can both win a fair amount, but each change can have negative consequences through the feedback. Agent 1 promises π if the other promises ρ and vice versa. The values $v_1(\rho/\pi)$

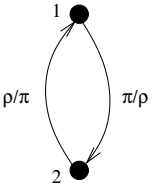


Fig. 15.3. Cooperative game, with conflict.

and $v_2(\pi/\rho)$ are inter-dependent.

15.2.3 Non-cooperative, no conflict

Two agents providing a service in a free market promise the same service to a third party, without any cooperation (fig. 15.4). The third party makes its choice and accepts the promised service partly from one and partly from the other. Each is promised a payment. Again, the value of the payment promises are quite independent, hence there is no conflict between agents 1 and 2.

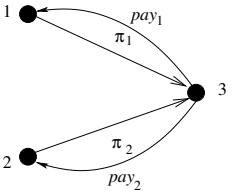


Fig. 15.4. Non-cooperative game, no conflict.

15.2.4 Non-cooperative, with conflict

This example is the classic and well-known Prisoner's Dilemma (fig. 15.5). Two agents promise to talk (or not talk) to their prison keeper (a third party). The prison keeper rewards both agents depending on the combined promises of both of the agents. The agents 1 and 2 are not in a position to cooperate (being

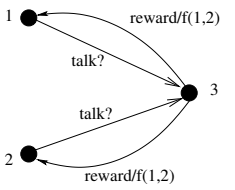


Fig. 15.5. Non-cooperative game, with conflict.

held in different cells), but they affect one another's outcomes indirectly. This is a conflict because the actions of each player can have consequences for both of them, and it is non-cooperative because the two players cannot communicate.

15.2.5 Constant-sum games

Let us now show that we can represent the special case of constant-sum (usually zero-sum) games, provided a certain level of cooperative infrastructure is in place. Such an infrastructure is often assumed in economic texts, but here we must be more careful. A constant-sum game is special because it implies an ability for one agent to control the value received by another agent. In a world of complete autonomy, that is forbidden, since each agent makes its own value judgements. Hence, to understand such a scenario, we must secure certain agreements between agents to agree on a currency standard. Constant sum games are not dependent on centralization, but we present two versions of such a game with and without centralization, both of which however show the fragility of the concept.

Theorem 5 (Constant-sum game) *A promise interaction that forms a constant-sum game requires a finite number (at least N) of agents (N players and at least one adjudicator which may optionally be taken amongst the players), and all agents must agree on a standard currency.*

Proof: Consider N promise interactions of the form:

$$\begin{aligned} n_i &\xrightarrow{\pi_i} A \\ A &\xrightarrow{r_i/\chi(\pi_1,\pi_2,...,\pi_N)} n_i. \end{aligned} \tag{15.3}$$

Each agent n_i promises an adjudicator A some service or behaviour π_i, and the adjudicator promises, in return, a piece of a common reward r_i, subject to a

constraint $\chi(\pi_1, \pi_2, \ldots, \pi_N)$. We would like this contraint to be that the sum of the values perceived by the agents is constant, i.e. we would like

$$\chi = \sum_{i=1}^{N} v_i(A \xrightarrow{r_i/\chi(\pi_1,\pi_2,\ldots,\pi_N)} n_i) = \text{const.} \tag{15.4}$$

However, each agent's valuation function is its own private estimate and is unknown to A, so the best it can do is to arrange for

$$\chi = \sum_{i=1}^{N} v_A(A \xrightarrow{r_i/\chi(\pi_1,\pi_2,\ldots,\pi_N)} n_i) = \text{const.} \tag{15.5}$$

Eqns. (15.4) and (15.5) can only agree if each agent n_i makes an agreement to use the same valuation $v_i = v_A, \forall i$, implying that the agents have a common currency. In a centralized framework, this implies an agreement of the form

$$\begin{aligned} A &\xrightarrow{v_A} n_i \\ n_i &\xrightarrow{U(v_A)} A \end{aligned} \tag{15.6}$$

between each node and the adjudicator. Hence the payoff matrix for the game is:

$$\begin{aligned} & U_i \left(v_i(n_i \xrightarrow{\pi_i} A, A \xrightarrow{r_i/\chi} n_i) \right) \\ = & U_i \left(v_A(n_i \xrightarrow{\pi_i} A, A \xrightarrow{r_i/\chi} n_i) \right) \\ = & U_A \left(v_A(n_i \xrightarrow{\pi_i} A, A \xrightarrow{r_i/\chi} n_i) \right). \end{aligned} \tag{15.7}$$

The adjudicator node has the role of book-keeper. There is no impediment to identifying A as any *one* of the n_i agents, the conditions are binding and sufficient regardless of which agent plays the role of A. Similarly, agents with the role of A can be repeated any number of times. ■

The issue of who elects the adjudicator of the game (game-master) is still open, and we shall not discuss that further here.

In the proof above, we made use of a centralized construction. This is not strictly necessary for the currency. It is sufficient that every agent or player participating in the game should share a common knowledge of the valuation standard for currency and obeys a constant sum rule. The knowledge of the currency standard can spread epidemically through a promise graph to achieve

the same result, provided the agents all abide by the same rules. This requires a special source node for the standard, which can be identified as A, so there is still a adjudicator node. The nodes must subordinate themselves to a set of rules that preserves:

$$\sum_{\pi_i,i,j} v_i(n_i \xrightarrow{\pi_i} n_j) = \text{const.} \tag{15.8}$$

This cannot be done without relinquishing part of their autonomy. Turning the argument around, it is likely that constant sum games do not represent a realistic interaction of autonomous agents.

15.2.6 Multiple strategies and mixtures

We have tacitly identified the promise of services with strategies in the interpretation of a game in normal form. A promise becomes an alternative in the expectation of rational play. However, there are subtleties about actions and promises to be addressed. What if, for instance, two promises are mutually exclusive? What happens when we form mixed strategies? There are two ways we can envisage mapping games onto collections of promises:

- Strategies are binary: either we keep a promise or do not keep a promise.
- Strategies are multiple: distinct alternative promises for future action, some of which might be mutually exclusive.

Consider the game matrix for a two-agent game below, with payoffs..

(1,2)	π_1	π_2	π_3
π_4	(10,5)	(4,6)	(3,2)
π_5	(0,5)	(7,2)	(5,4)
π_6	(10,5)	(6,6)	(8,3)

The normal solution concept for such a game is to search for an equilibrium configuration of strategies for the players. Generally, such equilibria required mixed strategies to balance their books. However, this leads to a difficulty in interpretation for promises. Some promises might be mutually exclusive, some promises might break other promises, etc. Thus we cannot simply search for a numerical equilibrium for this matrix without applying the logic of promises to the problem also. The formal ambiguity can be dealt with by making promises conditional on certain prerequisites. However, there will be many games represented in the literature that assume information channels that are unavailable to

autonomous agents. Readers should therefore take care to remember the implications of complete autonomy. In particular, two agents can only be assumed to share common knowledge about π if they agree[BFa]:

$$\begin{aligned} n_1 &\xrightarrow{\pi} n_2 \\ n_2 &\xrightarrow{U(\pi)} n_1. \end{aligned} \tag{15.9}$$

The issue of mixed strategies is something that must be addressed in both of the cases above. In a zeroth-level approximation to promise theory, one generally makes a default assumption that promises are always kept. This allows one to discuss the logic of promise graphs. However, as soon as one considers the matter of observation and repeated trials, and takes seriously the issue of why agents would keep promises, this default assumption becomes too simplistic to model real situations. We discuss some of these matters in the next section. For the remainder of the paper, we restrict out discussions to two-person games (or games that are two person factorizable).

15.3 Repeated bargaining: conditional promises

A dilemma that is faced by agents is whether to favour a long term benefit with a certain risk, or short-term guaranteed rewards by choosing a policy towards neighbours. If an agent only interacts once in its lifetime with another, it has nothing to lose by betraying the trust of the other. However, if there is the possibility of a future reprisal against it, the reckoning is a different one[Nas96].

Agents have the possibility of *trading* promises of different types using a valuation based on an agreed measure of common currency, as pointed out in ref. [BFa]. This allows them to secure long term relationships with one another in a way that secures stability of interaction. Selfish agents will only predictably continue to interact if they receive some net payoff from the interaction.

However, in a real world pervasive computing scenario, one must be careful in attributing the failure to comply with promises to non-cooperative behaviour, as one might in economics. There are several reasons why an agent might fail to live up to its committments in a cooperative relationship:

- Environmental uncertainties beyond the control of the agent intervene (noise).
- Unwillingness to cooperate emerges (change of policy).
- The failure of a dependency agreement, which invalidates a conditional promise (a third party spoils it).

This is not an extensive list, but each case is significant and important to the fragility of developing a regional policy in the ad hoc assembly of individuals.

So, given short term certainty versus long term risk, how might agents choose to behave? Suppose they surmise an incentive to cooperate:

1. Under what conditions would cooperation emerge and be stable in a group of selfish-minded individuals?

2. What kind of *behaviour* is it most beneficial for each individual to choose (strategy)? i.e. How does one harmonize selfish interest with common goals?

3. What might be done to promote the emergence of cooperation in a group by altering their policy for interaction?

The preferred solution method of such game theoretical problems is the concept of Nash equilibrium[Mye91]. An equilibrium is the end result of a 'tug of war' between players. Game theory therefore embodies a notion of *stability* in the result of the contest. This is an appropriate model for management. The restriction to two players simply means that, in a graph of interactions between individual agents, each negotiation is made between individual pairs, without reference to third parties. This is also what is appropriate for an autonomous environment. We recall some basic notions from game theory.

Definition 87 (Two person iterated game) *A* game *consist of two* players *or agents, which interact and make choices. For each choice i, by player a, and counter choice j by the other player, they each receive a payoff, which is determined by the* payoff matrix Π_a *with components* $[\Pi_a]_{ij}$ *(see fig. 15.6). The game can consist of one or several* rounds*. A game of several rounds is called an* iterated *game.*

This matrix form of the game represents the benefits and losses (hence risks) during *one* encounter between two agents. If they should meet again, and the previous encounter might have influence on how the agents act next time they meet, this is modelled by several *iterations* of the game. Each iteration hence represents one interaction between two agents, and this interaction could be defined in several ways.

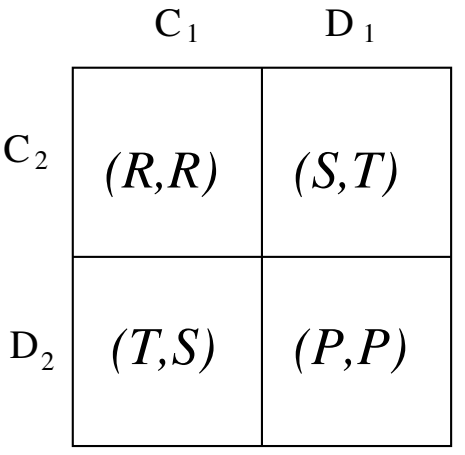


Fig. 15.6. The payoff matrix $\Pi_{(1,2)}$ representing rewards for a single move in a dilemma game. The components $[\Pi]_{ij}$ represent the four quadrants for $i, j = 1, 2$

Definition 88 (Move and strategy) *The choice of action* $\vec{M}_{a,i}{}^{\mathrm{T}} \in \{(1,0),(0,1)\} = \{\vec{C},\vec{D}\}$ *made by agent* a *in round* i *of a game is called a* move. *These concern keeping or breaking a given promise to the other player. For historical reasons, this is referred to as 'cooperation' (C) and 'defection' (D), see fig. 15.6. A complete sequence of choices* $\{\vec{M}_{a,i}|\forall i\}$ *for agent* a *takes over the entire game, constitutes the* strategy *of that player.*

Moves are in fact micro-strategies in the interpretation of the dilemma game as a normal-form, strategic game with constant payoffs. To avoid confusing micro- and macro-strategies, or extensive and normal form games, we use these terms: move and strategy.

To account for the effects of a receding history, the total payoff of a player in an interated game is generally a discounted sum of the payoffs in the individual rounds[Axe97, Axe84]. Let $M_{a,r}$ be the move made by agent a in the rth round of the game, and let Π_a be the payoff matrix in this round for agent (player) a, given a counter-player $\overline{a}$, then over m moves,

$$\Pi_{\mathrm{total},a} = \sum_{r=1}^{m} \delta^m \; (\vec{M}_{a,r}{}^{\mathrm{T}} \Pi_a \; \vec{M}_{\overline{a},r}). \tag{15.10}$$

The factor $\delta \leq 1$ is called the 'discount paramter'. One normally imagines oneself fixed at the present and looking out into (i.e. predicting) the future,

based on past experience, so that m recedes into the future. What happens in the distant future (or distant past) is less important than what is immediately upon us, so its importance to the payoff is reduced in weight, or *discounted*. This duality between past and future is the same as that in all probabilistic theories in which evidence from the past is used as a model for the future.

The dilemma game has proven to be a very powerful tool for describing co-operation. The goal of this method is typically to find stable *solutions* to the iterated game, by balancing short and long term gain against one another. The iterated game is played for a given number of rounds and the most profitable player, either during one round, or in the long run (several iterations) wins. The actual winner of a game is not really of interest to us in policy based management. What is important is the average equilibrium outcome, i.e. which strategies or choices of promises agents choose to keep. To be able to predict the outcome of such an iterated game, one needs:

- The strategies of the other players involved in the game
- The number of encounters/interactions with the other players
- The payoff/reward for the different combinations of encounters
- The start state.

This translates into finding out what kind of behaviour is most beneficial for an agent in the long run, dependent on what kind of environment the agent acts in, how often each agent meets agents with certain behaviour and so on.

15.4 Promises: constructing policy

It is plausible that there is a connection between the two person games studied in the theory of cooperation and the likelihood that promises are kept between individuals in a pervasive computing environment. We shall now show this.

A promise graph $\Gamma = \langle I, L \rangle$ is a set of nodes I and a set of labelled edges or links between them L. Each directed edge represents a promise from one agent to another. The set of links L falls into several non-overlapping types, representing different kinds of promise. Thus each promise begins with a label π_i, denoting its type, and with a specification of the promise constraint, which it promises to uphold.

Lemma 2 (Games are bilateral promises) *Let a two person strategic game be represented by a pair of* 2×2 *matrices* Π_a*, and moves* $\vec{M}_i$ *where* $n_i, i = 1, 2$ *are*

the agents and $a, b = 1, 2$ label the choices to cooperate or defect in the payoff matrix, e.g. $[\Pi_i]_{ab}$. There is a mapping of each game between the players onto a pair of promises.

Proof: For the players in the game, introduce a promise of type π_a and a return promise of type π_b, which are traded in the manner of fig. 15.7a. Now let $\vec{T}(\cdot)$ be a 'truth' function that maps any promise into a one of the values $\{\vec{C}, \vec{D}\}$, indicating whether the promise is obeyed or not by the promising agent. The truth function may be associated with a move in a game, of a type that maps one to one onto the promise type, and hence may be written equivalently:

$$\vec{T}(n_1 \stackrel{\pi_i}{\to} A_2) \to \vec{M}_1 \tag{15.11}$$

$$\vec{T}(n_2 \stackrel{\pi_j}{\to} A_1) \to \vec{M}_2 \tag{15.12}$$

$$\left[v_a\left(n_1 \stackrel{\pi_i}{\to} A_2, n_2 \stackrel{\pi_j}{\to} A_1\right)\right]_{ab} \to [U_i]_{ab} \tag{15.13}$$

where v_i is the matrix-valued valuation function for agent i, which interprets the value of the mutual relationship to agent i; π_a represents the promise body, which is labelled by the types a, b. We must have either $(A_1, A_2) = (n_1, n_2)$ or $(A_1, A_2) = (n_3, n_3)$ to mediate the game. The proof is now trivial noting that the arrow is from promises *onto* games, by a policy determined function v_i. ■

Note that, although we can identify a game with a pair of promises, not all promises can be meaningfully viewed as games, as we see in section 15.6. Any reasonable function v_i can be used to defined the mapping from promise to game, since the function v_i is to provide a autonomous *valuation* of received promises by each agent n_i. The game can then be used as an estimator for the rational behaviour of the agents as they interact.

A consequence of this relationship between promises and games is that promises must also specify a policy for stabiliizing the cooperative relationship over time, i.e. a strategy for the entire iterated game.

15.5 Management dilemmas

A common approach for modelling and analysing cooperation is to use dilemma games from two-person game theory[Axe97, Axe84]. Here two individuals, without strictly opposing interests, have the possibility of exploiting each other's willingness to cooperate for selfish gain. This is a scenario that has shown to fit several real-life cases[Axe97, Axe84]. It is known as a cooperative dilemma.

In the dilemma game, each player can choose between two different moves, *cooperate* (keep promise) or *defect* (i.e. do not keep promise). The interpretation of the dilemma game symbols is thus: T is the payoff for receiving services from another agent, while refusing to contribute; R is the reward two agents receive when both are cooperating; P is the inconvenience both agents experience when they both refuse to collaborate; and S is the inconvenience of servicing an agent who refuse to contribute anything back.

There are many possible outcomes, which depend on the relative sizes of these parameters. For the game model to qualify as a Prisoner's Dilemma, in the common usage of the term, certain requirements need to be fulfilled: (i) $T > R > P > S$, and (ii) $R > \frac{(T+S)}{2}$. These requirements ensure that conclusions associated with the game are true.

A player can win the maximum payoff by 'defecting' (failing to act according to its own promises) when the other player is 'cooperating' (acting according to its promises), but if both players are defecting, they both receive a lower score than if they both had cooperated. Constraint (ii) assures that mutual cooperation is more beneficial than alternating cooperation and defection between the two players.

What makes this game model interesting, is the controversy that even though the choice *defect* is *dominant*, both players will actually end up with a better result cooperating in the long term. This model is completely analogous to a scenario of autonomous agents, looking to establish agreements (promises) between one another, in order to create a larger system voluntarily. There is a potential long-term reward associated with providing services for other peer agents, but avoiding a commitment could thus seem beneficial in the short run.

We model this situation with multi-move games in which one agent can follow the history of its interaction with another and *punish* previous defections from the other player by failing to behave cooperatively in return. An example strategy, which has received much attention in the literature, is *tit for tat*. This has proven to be very successful in playing cooperative dilemmas with a good average performance. Tit for tat is a macro-strategy and it is defined as follows:

Definition 89 (Tit for tat) *Let r label a sequence of rounds in a game, and let $\vec{M}_{1,r} \in \{\vec{C}, \vec{D}\}$ be the move taken by agent 1 in round r, etc. The tit for tat strategy is defined by:*

$$\vec{M}_{1,1} = \vec{C} \tag{15.14}$$

$$\left.\begin{array}{l} \vec{M}_{1,r} = \vec{M}_{2,r-1} \\ \vec{M}_{2,r} = \vec{M}_{1,r-1} \end{array}\right\} r > 1. \tag{15.15}$$

Notice that the agents always start out cooperating*, or obeying policy rules, and then do against the opponent what the opponent did on the* previous *move.*

The success of this strategy, in dilemma games, is characterised by the fact that it gets high scores on average, even though it does not necessarily always win (i.e. it does not always maximize the utility of its interaction). This strategy has several beneficial qualities: it is simple, it rewards cooperation, it punishes defection and it is forgiving, hence single defection noise (including accidents) does not harm a player too much. The disadvantage is that it does not exploit recovery from random noise well enough. It tends to end in fruitless reactive oscillations between agents, once they have started. A continuous generalization of the game has further instability problems[BBJF]. Some thing is required to dampen out the oscillations caused by random errors. A small amount of noise to unleash oscillations throughout a peer network, in which the agents had dependent promises.

The real dilemma in these games is this: if an agent in a trading relationship does not retaliate to broken promises, it will allow itself to be exploited, exhausted and stability will be compromised. However, if it does retaliate, it participates in the contributing to a policy instability in the network.

15.6 Examples of games and the economics of promises

What is the complete relationship between games and promises? We need to establish how the strategies interact and affect one another. Some relationships are purely fortuitous, with no guarantees of receiving a payoff for their actions. Others interactions are games which are bound by a trade of mutually beneficial services. Thus not all promises are games, but all dilemma or bargaining games involve promises. How does the probability of keeping a promise relate to the experience gained from iterative dilemma games, and the discount parameter?

Since the cooperative dilemma gives us a natural notion of stable equilibrium

for autonomous agents, let us ask: in what sense is a promise a cooperative dilemma, and which agents fall outside this model?

Consider two nodes, node A and node B, which each represent an autonomous node. As described in [BFa], a promise $a \xrightarrow{\pi} b$ from a to b is represented in graph notation by a directed edge (arrow) from node a to node b, and the label on the edge defines the *content* of the promise.

Definition 90 (Policy dilemma) *A bi-lateral promise between two nodes, in which cooperation is interpreted as* keeping *the promise, and act as promise, and defection is defined as not keeping the promise. In order to qualify as a bargaining game, both nodes must receive some utility from the bilateral arrangement.*

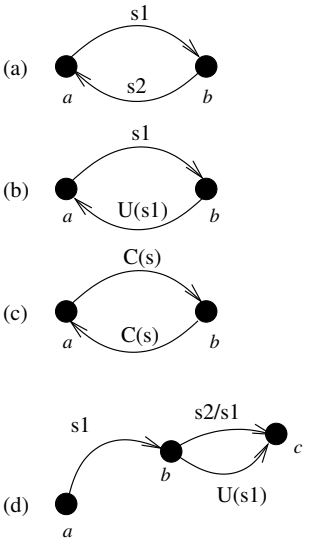


Fig. 15.7. Some 2 person games and their corresponding promises.

Consider the images in fig. 15.7. Here we see a number of promise graphs, some of which are policy dilemmas and some of which are not. There are three basic categories of promise[Bur05]:

1. In a service promise of the form $a \xrightarrow{s} b$, node b is the recipient of something of value (the promised service).

2. In a use-promise, $a \overset{U(s)}{\rightarrow} b$, node a promises node b that is will avail itself of a service s , which is has been promised by some node. In this case, node b does not receive anything tangible, except an assurance that a will use the service it has been offered.

3. Similarly, in a cooperative promise $a \overset{C(s)}{\rightarrow} b$, node a promises node b that it will imitate b's behaviour with respect to service s. Again, the only thing b receives is an assurance.

We must now ask: are all these dilemma games? Consider fig. 15.7.

In case (a) the two nodes exchange services or other similar promises of some value to one another. We may consider this interaction as representing a barain-ing game between the players, if the promises are judged beneficial to the agents, e.g. BGP peering agreements between service providers.

In case (b) however, the assurance of usage does not seem enough to bargain with. Indeed, node a says "if you do not give me s, I will not use it!". This does not seem like a strong bargaining position. One could similarly apply (b) to a cooperative promise: "If you do not give me s, I will not do as you do with regard to s" (give myself s). Again, there are no grounds for bargaining. This could be misleading however. Consider the example of refuse recycling or garbage collection in society. Users promise to deliver their garbage and the collectors agree to take it. Both parties can benefit from this because by *not* cooperating they would incur a loss. This warns that caution is required in jumping to conclusions.

In case (c), there is a game, since the two promises are on equal footing: "if you don't do as I do, I won't do as you do". In both cases, the promises are of equal value and so they have a symmetry of mutual assurance, and we can define them to be a game. Indeed, case (c) can easily be shown to be a stable fixed point.

Case (d) contains no game either. Node c gives nothing in return to node b for its service s_2; node b offers nothing in return to node a for s_1.

The services are promised without any trade taking place. In order to establish a game, an agent must be able to exert an influence against its peers. It must have something to bargain with. There are two kinds of nodes that do not fit into this contract of stability, easily located by network analysis.

Definition 91 (Opportunist sink) *A network sink (a node with out-degree zero) is* opportunist. *It is vulnerable to loss of service, as it offers nothing in return. It is a non-profit node.*

Definition 92 (Altruist source) *A network source (a node with in-degree zero) is an* altruist*. It is vulnerable and exploited, as it gives without taking anything in return. It is a 'single point of failure' for the dependent promises.*

Thus network analysis could enable us to repair the lack of economic incentive to cooperate in case (d) by modifying the graph to allow for some chain of payment (see fig. 15.8). Thus longevity of relationships provides a direct motivation

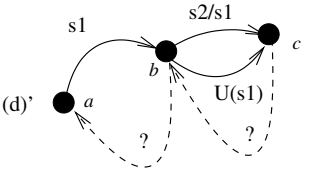


Fig. 15.8. An act of uncertain altruism can be turned into an incentivized bargaining game by assuring some sustaining payoff to the altruist.

for introducing a trading currency.

15.7 Minimum incentive requirements

We can now propose some minimal requirements for sustainable promises, based in economic incentives.

Proposition 1 (Common currency) *The need to bargain or trade with different currencies motivates an undirected graph of common currency, representing bilateral promises.*

Proposition 2 (Principle of sufficient remuneration) *Every agent should receive a 'handshake' promise in return for giving a promise. The handshake can be valued in any currency used by the agent.*

This proposition can easily be proven if one posits equilibrium as a criterion for a well-formed policy. We shall not attempt to squeeze this into the present work.

Example 46 (Legally binding agreements) *The literature of games and economics often refers to 'binding agreements' in the sense of agreements that are enforced by some legal framework[BG02]. In promise theory there is no such outside legal framework, unless one builds it from promises. Such a framework is very complex. A third party (government or law court) must promise to make laws known. All agents must promise to use the law. They must promise to make visible any contracts and practices they make to an arbitrator, who in turn must promise to inform law-enforcers of any breakage of laws. These in turn could enforce an isolation (incarceration) of an agent. To make an agreement binding by force, one must subordinate oneself to an outside force, or suffer large consequences. Such a scenario takes considerable infra-structure, which shows that commonly held assumptions can often represent very complex scenarios (which is why criminals stand a chance of succeeding in breaking laws). Ironically the main reason for contract success is a sound economic basis, not the threat of punitive action.*

The idea of offering different interpretations of currency has been considered also by Piaget[Pia95]. In his view, societies of agents bind into working constellations because they value one another by different criteria. Consider a transaction performed between two 'agents', represented by A and B, where A performs some action *on behalf* of B. Piaget assumes the following basic conditions, which differ from our starting point:

- A common scale of values for all agents.
- Conservation of values in time.

and remuneration values: i) Renouncement value r_A, representing the *investment* made by agent A in order to perform an action on behalf of B. ii) Satisfaction value s_B, representing the satisfaction expressed by B as a result of receiving the action from A. iii) Acknowledgement value t_B: representing the acknowledgement made by B to A as a result of receiving the action. iv) Reward value v_A, representing the associated value of receiving the acknowledgement from B, as experienced by A. Piaget distinguishes between so-called *real* and *virtual* values, where r and s are the real values, and t and v the virtual values in the transaction. The virtual values turn into real values when they are 'cashed in'.

15.8 Service Level Agreements as promise networks

Will service trade be *stable* over time? Consider a graph of n nodes, each making promises to a subset of the other nodes. Each promise is represented by

an arrow between the node making the promise, and the node which is the receiver of the promise. In the result graph, the number of directed edges indicates how many *relationships* are established in the group, which also says something about how *clustered* the group is. Using graph theoretical terminology, we can compute the *connectivity* and the *clustering coefficient* of the group[Bur04a, Bal97, Ber01, Wes01].

If we, for instance, assume all nodes are trading *tit for tat*, and the environment is without noise, it is possible to determine, without uncertainty, what choices each node will make in the next iteration. This means we can determine which nodes will cooperate, or how many will cooperate.

If we can only speak of probabilities, the probability of whether each promise will be kept, is then a number in the interval $[0, 1]$, which in graph theoretical terminology is a *weight* on the corresponding promise edge. For two agents, we can easily predict the stable configurations under tit-for-tat by computing the spanning eigenvectors of the common currency promise graph. To do this, we write the strategy as an iterative matrix operator whose fixed points will be the stable solutions of the policy:

$$\lambda \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} \tag{15.16}$$

which has stable eigenvectors

$$\begin{pmatrix} M_1 \\ M_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{15.17}$$

This shows that the two strategy modes that can persist in the graph are that both agents agree to agree (either keep their promises or not) for all time, or they enter into an eternal battle of opposites (cooperation and defection) for all time. In the absence of an initial condition (as in def. 89) there is no way to pick out which of the former is the case, but it seems reasonable to assume that promises will initially be kept if an agent bothers to make them. This method can be extended to larger graphs and we shall return to this matter in later work.

To choose the stable alternative, we must ensure sufficient promises to fulfill all of our propositions for stable cooperation.

Let us define an *agreement* as a bilateral promise between two agents to affirm and comply with a common statement. A *contract* is a bilateral collection of promises and commitments between the agents. We then have that a contract Service Level Agreement (SLA) is a mutual promise between agents to comply with documented promises that go in both directions. Our results show minimum requirements for such promises, for successful service management.

15.9 Summary

In this paper we have considered the relationship between promise graphs and the management of services through voluntary 'economic' cooperation. This viewpoint is particularly applicable for peer to peer services and free market economic models for electronic commerce. We have shown that, given the ability of agents to make personal valuations of promised services, there are natural promise configurations that form economic games. We propose that *sustainable configurations* are naturally predicted by the outcomes of these equivalent games, and that imbalanced promises are likely to lead to disintegration of promise networks, or unsustainable Service Level Agreements.

The existing body of work from cooperative game theory suggests that a complete view of a human-computer policy, as an equilibrium of individual promises, requires three things: identifiability of agents, a notion of common currency for motivating cooperation, and a stabilizing principle of remuneration (trading), that invokes bilateral promises to bind all nodes together. The fragility of promise networks is easily analyzed by methods of graph and reliability theory.

Naive cooperative game theory suggests policy strategies for maximizing the utility of an autonomous agent's policies.

1. *Initial cooperative behaviour:* When establishing a new promise or relationship, an agent should always act in a cooperative way, i.e. should keep the promise made to another agent.

2. *Punish bad behaviour:* If an agent fails to fulfil/keep a promise, this should be punished by the environment, so the agent lose some privileges.

3. *Reward good behaviour:* Agents that keep their promises should be rewarded.

This viewpoint must deal with defections that happen by accident: i.e. faults or 'acts of God'. Immediate punishment is an efficient, but not necessarily desirable strategy, unless used as an incentive to motivate redundancy and fault tolerance of systems. Simulation leading to empirical conclusion is an avenue for future study.

Pervasive computing scenarios typically imagine an extremely dynamic environment, where the number of agents involved changes. Also, they are assumed to be very *heterogeneous*, which would imply a wide variety of different promises. We propose that understanding the global impact of this world requires an economic model. There are many threads to pick up on. We hope to report on these in future work.

Notes

[48]In physics the bindings are motivated by energy considerations. Energy is the economic currency of physics.

Part 3
Appendices

Appendix A
Promises, obligations, commands and requests

Throughout this book, we use the term 'promise' with a more or less constant intent. Nevertheless it turns out to be difficult to provide a clear definition without making it unreadable and sacrificing intuition. Thus, in this Appendix, we would like to expand on the manner by which we arrive at our definition 14 of chapter 3 n commitand its variations by extending the philosophy of chapter 1.

A.1 Ontology

We proceed from the following:

Definition 93 (Promise) *A public announcement (made by an agent called promiser and performed as an act of free will) of an intention, of a plan to be executed or of the expectation of a fact to hold true either in the past the present or the future (the promise body usually composed of a type or quality and a size or quantity). Among the audience (scope) of the promise there may or may not be one or more specifically designated agents who are the intended beneficiaries of the fact or action implied by the body. It is conventional for the announcement being classified as a promise is that it should effect expectations within the promisees which are positively valued as such.*

For use in engineering, computing and other technical disciplines, where promisers and promisees are likely to be inanimate agents, the following weakened forms of this definition will be relevant:

1. A promise may not have promisees, in which case it is merely an announcement of a plan or of an expected or existing state of affairs to the agents in its scope.

A promise to a promisee has in addition to being an announcement of a plan or state of affairs the aspect that the promiser is aware of the promisees interest in the promise being kept, as well as the fact that the promisee may expect that this awareness is seriously taken into account when the promiser decides not to keep the promise.

2. A promisee's changed expectations, on receiving a promise, need not be made explicit and the change need not be made in a positive direction.

   A threat is a promise made with generated expectations valued negatively rather than positively. Thus threats are included in promises, an conversely for that matter.

3. There may be different promisees, and these may value their change in expectations differently. What is an attractive promise to some may be a threat to someone else.

4. Rather than speaking of an act of free will, both issuing the promise and acting in compliance with its body should be properly classified as *autonomous behaviour* of the promising agent.

Below we will provide two weaker definitions of promises (that is, by qualifying more kinds of announcements as promises), and definition 95 will be the final one for our purposes.

A.1.1 Notions related to promises

Lots of notions can be found that relate to the notion of a promise. Here is a listing.

- Obligation
- Promise
- Forced promise
- Autonomous promise
- Command (instruction)
- Request (an asking)
- Suggestion to perform a plan
- Announcement of plan

- Announcement of intention
- Plan (intention of plan, intention to execute plan)
- Intention
- Execution of plan (activity)
- Forced activity
- Compulsive activity

Besides these concepts that all have to do with speaking about plans and their execution either by the agents themselves or about other agents there are surrounding notions that come into play.

- expectation
- hope
- fear
- ambition
- conventional pattern of action
- will (free will)
- trust
- distrust
- positively valued expectation
- negatively valued expectation
- happy state of an agent
- partially happy state of an agent (positively valued aspect of state)
- unhappy state of an agent
- partially unhappy state of an agent (negatively valued aspect of state)
- agent life-cycle with at least the stages: birth, active phase, and death
- force exercised upon an agent
- objective

All of these notions may play a role when describing a promise. That makes the concept particularly complex. Our definitions of promise take into account only a limited set of aspects.

A.1.2 Why make promises

When working towards a definition of promises some examples are needed where the act of promising is convincing form of a rational behavior. Interestingly one can find examples where promise making cannot be replaced by other actions. Here is an example.

Posthumous publishing Suppose that agent B is approaching the end of his life and B is aware of that fact. B asks agent A to take care of publishing his nearly finished book via publisher E where all that needs to be done before submitting is one round of proofreading.[1]

We assume that B has no life after death, all goodies relevant to B happiness must materialize during his life. The following scenario is now plausible:

- We assume that B has no means available to arrange via his will that someone takes care of publishing the book, and that B has no access to other authorities who may force A to publish the book after B's death by means of an immutable obligation for A to do so.
- In response to B's request, A promises to B to proofread the manuscript, to repair obvious mistakes and to submit it to publisher E subsequently.
- As a consequence of hearing A's promise B reaches a partially "happy" state (regarding the expected fate of his most recent manuscript) because of the positive expectation that the promise will in fact be effected.
- The 'happy' state of B is a positive value even if it will not lead to any action observable by other agents. In other words it is to be valued positive as such that B ends his life while being in a partially 'happy' state. (The more "happy" the better so to speak.)
- B has no possibility to check or test whether or not A will keep its promise,
- For B there is no other way to obtain this particular state of happiness before the end of his life than to ask for that promise to be made by someone who he can communicate with and in who he has sufficient trust.

Modifications can be easily imagined. In this case, however, the promise made by A seems not to produce any good over and above what would have been achieved if A made an announcement of plan with B in the scope of that announcement.

[1] A classical related example is Vergil, who asked to be promised that his manuscript Aeneid be destroyed.

- In a modified example where B is still enabled to edit his will B may try to force A into making the promise by making the execution of part of his will towards A dependent on that.

- In yet different circumstances B may try to create an obligation for A to execute the promise body by asking some higher authority to issue a command towards A for doing so.

- Assuming that after his death B is insensitive to any further event, (a consequence of the assumed absence of life for B after his death) the thereafter merely A's happiness is at stake if he fails to live up to his promise. If more agents were in scope of the promise A may have additional incentive to live up to it.

- This example provides an example of a promise where the objectives of B cannot be reached by anything else than a promise being made. A's mere announcement of plan will generate the same expectation inside B and for that reason it qualifies as a promise in the sense of definition 93. None of the other related notions appropriately describe what is going on. This example may be considered a proof of existence of the concept of promise as defined in 93.

Feedings a cat during its owner's vacation Agent B will be on vacation for one month and needs his plants watered. Unless this is properly done the plants won't survive. B asks A to do so and A makes the promise to do so with only B it scope of the promise.

Now like in the previous example A cannot force B to live up to the promise during the period of his vacation. But there are significant differences: (i) after returning B can see whether A has done his job, (ii) if A has not lived up to expectation B may try to punish him for not keeping his promise (it is quite difficult to formulate in abstract terms that B does neither have the power nor the intention to do so), (iii) after his trip B can reward A for keeping his promise (and by making the promise A may develop the positively valued expectation that this will take place), (iv) B might return sooner and see what is going on (if A is not keeping his promise), (v) B might lose his trust in A while being on vacation and for that reason ask another agent to see into the matter.

There seems to be no doubt that all these differences do not change the situation to such an extent that B's announcement of plan fails to qualify as a promise. The reason for that is that under default circumstances B is dependent on A keeping is promise as an act of free will. Only if additional changes in

the situation emerge the setting is changed in such a way that what started its life as a promise made autonomously as an act of free will turns into the forced execution of a plan against one's free will. Here we see that when checking that executing the promise body should be an act of free will a default logic may come into play.

A.2 Derived constraints on promises

1. The absence of both obligation and force is essential for classification of an announcement of plan as a promise. Thus either the notion of free will is take as a primitive one or it is reduced to the absence of force and obligation. Free will being a complex concept itself its seems compelling to view both force and obligations as simpler and to replace in the definition of promise the free will requirement by a requirement that no force or obligation are relevant for the announcement or the execution of the plan in its body. This would be

   Definition 94 (Promise) *A promise is defined as in definition 93, but now with the free will requirement replaced by the requirement that force or obligation are not involved.*

2. If A is forced to execute plan P he cannot promise to do so at the same time (unless P is unaware of the force applied). The same holds for making the promise. Here we arrive at an unclear area: suppose A is obliged to perform plan P but is unaware of that fact. The A can be asked to promise to do so. Has A made a promise. According to both definitions he is not. But he cannot see this fact for himself. Should we take into account the possibility that A cannot reliably judge whether he is making a promise when he subjectively thinks he is doing so. This is less attractive and therefore we are led towards a further sharpening of our definition:

   Definition 95 (Promise) *A promise is as defined in definition 93 but now with the free will requirement replaced by the requirement that to the best of A's knowledge force or obligation are not involved.*

   This definition leaves open the possibility that B asks A to promise P while B knows that he can force A to do so and B is unaware of that fact. Still, according to definition 95 the announcement qualifies as a promise for B.

This subjective aspect reinforces the reduction of 'free will' to the absence of force and obligation. That can be easily weakened to the perceived absence of both. Instead the distinction between free will and perceived free will is harder to grasp.

A.3 Force, command and obligation

A request is complementary to an announcement of plan. A request by B to A can be issued and it is reasonable that B provides either an announcement of plan or a promise in its return. For an announcement of plan it is no requirement that force or obligation are absent or that any form of free will is established or assumed by either by A or by B or any other agent involved.

An obligation differs from a request in the sense that the obligation for B to perform plan P should be derived from a general rule that agents which match properties X should perform in accordance which Y and that for B acting in accordance with Y implies to execute plan P.

Obligations (for A) can exist without having been announced by any agent. A typical rule for creating obligations may be that A should perform in compliance with the commands issued by B (because A is working under B's supervision). This turns all commands issued by B into obligations of A. At the same time it makes implausible the qualification as a promise of any announcement of plan made by A towards B, insofar as these plans are with in the scope of B's authority over A.

Force is an other matter altogether. Force is not derived from a general rule but it comes into effect in specific circumstances. It may be questioned whether or not a human agent can ever be forced into any action at all. In the setting of engineering, e.g. computing, force is more easily imagined. An agent may be forced if some of its actions are given a very high priority in some context. But will assume the existence of force in the more general case.

Coercion, a kind of force, is in place if an agent is made aware of negative consequences unless some plan is performed. These negative consequences lead to a state of unhappiness which is valued negatively. Indeed the example of a promise given above can be extended with a historic context in which B's death is expected because he has not complied (which may very well have been an act of free will) with some command (for instance by A) and has been awarded a capital punishment for that reason. Still B's unhappiness about the state of affairs can be diminished by the promise made by A concerning finalization of B's book. The qualification of B's announcement to see to its publication as a promise is unaffected by the context in which an attempt to apply force led B

into the problems that provide an incentive to ask for A's promise.

A.4 Another promise

Suppose at the end of B's life, A makes B the promise that A will have a further life in another world. This generates a positive expectation inside A's mind and he ends his life in a 'happy' state for that reason. Assuming that one does not believe in the second life in another world one may still acknowledge the state of happiness thus created for B.

Are we to conclude that: (i) the announcement of future fact made by A qualifies as a promise, (ii) because of the positive valuation by B of the announced fact that generates a state of happiness which is for that reason to be valued positive as well the very act of making the promise can be evaluated positively as well.

This is a difficult matter. A promise is qualified as a deception if the promiser does not expect either the announced fact to hold true or the announced plan to be performed (depending on the form of the promise). As it stands the act of announcing a deceptive plan or fact is not to be valued positively as such in spite of the positive value assigned to it by the promisee. The matter is hard to resolve: if announcing eternal life in another world is the only possible way to increase B's happiness at some stage, should A for that reason do so? Perhaps some people adhere to a specific faith only for the reason that when this case arrives they can make the promise without hesitation. These decisions are an ethical matter.

Nevertheless it seems defensible to exclude from qualification as promises those announcements of plan or fact for which no active (living) active agent, either now or in the future can ever assert whether the announced fact has come about or whether the announce plan has been performed. The the announcement of eternal life is not qualified as a promise but should be termed differently, e.g. as a 'religious promise', which fails to qualify as a promise just like a cancelled flight qualifies as a flight.

In this way deceptions still qualify as promises in as far as the coming about of the announced fact or plan can be determined objectively in some time to come. This in turn can be taken as incorporated in the notions of plan and fact rather than in the definition of a promise as such. Thus the event of B's eternal life in a second world would not be regarded an admissible 'fact' (but say merely a religious fact instead), thus allowing not to classify A's announcement of B's life after death as a promise.

Again one might criticize this distinction because it is unknown now which

forms of knowledge future generations will have available to them. Perhaps some day the death will appear to the living so that these must acknowledge their preservation after death. But now one may use default reasoning to argue that sources of knowledge are absent unless known to be present.

A conceivable argument, e.g. that future generations will develop a 'religious radar' which enables them to spot the deceased in heaven cannot be refuted by any means, but default reasoning indicates that it should not be taken into account unless it has actually materialized. Such an argument cannot be used to qualify a supposed state of affairs in a promise body as a potential fact which is necessary for the qualification of the announcement as a promise.

A.5 Conclusion

We have shown to our satisfaction that promises exist. However, to arrive at an applicable definition, the concept must depend on that of force and obligation, in the sense that the absence of these must be required. We conclude for that reason that in a detailed analysis of the general concept of a promise, it cannot precede the introduction of any notion of force and obligation.

However, a drastic simplification is achieved if the concept of autonomous action is used as a primitive instead. Now the absence of force or obligation is implicit in the notion of *autonomy*. It seems reasonable to use the following definition of a promise in the case of computing.

Definition 96 (Promise) *A promise is as defined in definition 93 but now with the free will requirement replaced by the requirement that act acts autonomously both when making the announcement and when performing the plan implied by its body if any.*

When using this definition in engineering, e.g. computing, care should be taken that when notions of agent involved become more refined, at some stage, autonomy ceases to be a primitive notion and the more restrictive view such as in definition 95 should be preferred.

Under this assumption its seems valid to consider a promise according to definition 96, as a concept that is independent of the concepts of force and obligation.

Appendix B
Expectation

B.1 Defining expectation

Here we expand on our notion of an expectation function for completeness. These details are not essential to the arguments in the rest of the paper.

An expectation function is a statistical concept that relies either on a body of evidence, or alternatively on a belief informed by limited observation. Such evidences or beliefs are summarized by a probability distribution over the different possible outcomes. We shall consider mainly the outcomes "promise kept" and "promise not kept", though varying degrees are possible.

The notion of an expectation value is well known from the theory of probability and can be based on either classical frequentist-probability or Bayesian belief-probability[GS01]. There is, for this reason, no unique expectation operator.

Why dabble in intangibles such as beliefs? Computer systems are frequently asked to trust one another without ever having met (for example when they automatically download patches and updates from their operating system provider, virus update from third-parties or even accept the word of trusted third parties in identification) – thus they have little or no empirical evidence to go on. Each time they interact however, they are able to revise their initial estimates on the basis of experience. In this regard, a Bayesian view of probability is a natural interpretation, see e.g. [Pea88]. This is a subjective view of probability that works well with our subjective agents.

Definition 97 (Expectation function $E(X)$) *Given random variables X, Y, an expectation operator or function has the properties:*

1. *If $X \geq 0$, $E(X) \geq 0$.*
2. *If $a, b \in \mathbf{R}$, then $E(aX + bY) = aE(X) + bE(Y)$.*
3. *E(1) = 1.*

For a probability distribution over discrete classes $c = 1 \ldots C$, it is the convex sum

$$E(X) = \sum_{c=1}^{C} p_c X_x \quad \Big| \quad \sum_{c=1}^{C} p_c = 1. \tag{B.1}$$

The expectation value of a Bernoulli variable (with value 0 or 1) is clearly just equal to the probability of obtaining 1 $p_1 = Pr(X = 1)$. In general, a promise might lead to more than one outcome, several of which might be acceptable ways of keeping the promise, however this possibility only complicates the story for now, hence we choose to consider only the simplest case of

Definition 98 (Agent expectation $E_A(a \xrightarrow{b} c)$) *The agent expectation $E_A(X)$ is defined to be the agent A's estimation of the probability that a promise $a \xrightarrow{b} c$ will be kept.*

This can be realized in any number of different ways, e.g. as a mapping from an ensemble of evidence of size N (with the binary outcomes 0 and 1) into the open interval:

$$E_A : \{0, 1\}^N \to [0, 1] \tag{B.2}$$

or it could could be an ad hoc value selected from a table of pre-decided values.

B.2 Ensembles and samples of evidence

An ensemble is a collection of experiments that test the value of a random variable. In one experiment, we might evaluate the agent expectation to be p_1. In another, we might evaluate it to be P_1. What then is the probability we should understand from the ensemble of both? We expect that the appropriate answer is an average of these two values, but what if we attach more importance to one value than to the other? Probabilities discard an essential piece of information: the size of the body of evidence on which they are based. Let us consider this important point for a moment.

B.2.1 Frequentistic interpretation

In the frequentist interpretation of probability, all estimates are based on past hard evidence. Probabilities are considered reliable as estimators of future behaviour if they are based on a sufficiently large body of evidence. Let lower-case $p_1 = n_1/n$, the probability of keeping a promise, be based on a total of n measurements, of which the *frequencies* n_1 were positive and n_0 were negative, with $n_1 + n_0 = n$. Also, let upper-case $P_1 = N_1/N$ be an analogous set of measurements for which $N \neq n$. How should we now combine these two independent trials into a single value for their ensemble?

In the frequentist interpretation of probability, the answer is clear: we simply combine all the original data into one trial and see what this means for the probabilities. Rationally, the combined probability E for the ensemble must end up having the value $E = (n_1 + N_1)/(n + N)$. If we express this result in terms of the probabilities, rather than the frequencies, we have

$$E = \left(\frac{n}{n+N}\right)\ p_1 + \left(\frac{N}{n+N}\right)\ P_1 = \frac{n_1 + N_1}{n+N} \tag{B.3}$$

This leads us to the intuitive conclusion that the probabilities should be combined according to a weighted average, in which the weights are chosen to attach proportionally greater importance to the larger trial:

$$E = \alpha_1 p_1 + \alpha_2 P_1, \quad \alpha_1 + \alpha_2 = 1. \tag{B.4}$$

In general, then, with T trials of different sizes, the result would be a convex combination of the expectations from each trial:

$$E = \sum_{i=1}^{T}\ \alpha_i p_i, \quad \sum_{i=1}^{T} \alpha_i = 1. \tag{B.5}$$

In the case that there are more possible outcomes than simply 0 and 1, the same argument applies for each outcome.

The problem occurs when we do not have complete knowledge of the sample sizes $n, N, \ldots$ etc., for, in this case, we can only guess the relative importances α_i, and choose them as a matter of policy. If, for example, we could choose to make all the α_i equally important, in which case we have no control over the importance of the expectations.

This so-called *frequentist* interpretation of expectation or probability generally requires a significant body of evidence in the form of independent events to generate a plausible estimate. However, in most ad hoc encounters, we do not

have such a body of evidence. Trust is usually based on just a handful of encounters, and one's opinion of the current evidence is biased by prior expectations. Hence, we turn to the alternative interpretation or Bayesian probability.

B.2.2 Bayesian interpretation

The policy formula in eqn. (B.5) is essentially a Bayesian belief formula, which can be derived from the classic Bayes interpretation for *a posteriori* belief.

Suppose we devise an experimental test e to determine whether a hypothesis H of expected trustworthiness is true. We repeat this test, or borrow other agent's observations, thus collecting n of these $e_1 \ldots e_n$. The result for $P(H|e_n, e)$, our belief in the trustworthiness-hypothesis given the available evidence, changes by iteration according to:

$$P(H|e_n, e) = \frac{P(H|e_n) \times P(e|e_n, H)}{P(e|e_n, H)P(H|e_n) + P(e|e_n, \neg H)P(\neg H|e_n)} \qquad \text{(B.6)}$$

where we feed back one value $P(H|e_{n-1}, e)$ from the previous iteration as $P(H|e_n)$, and we must revise potentially two estimates on each iteration:

- $P(e|e_n, H)$ is our estimate that the test e will show positive as a direct result of the Hypothesis being true, i.e. because the host was trustworthy.
- $P(e|e_n, \neg H)$ is our estimate of how often e is true due to other causes than the hypothesis H of trustworthiness, e.g. due to trickery.

Note that $P(\neg H|e_n) = 1 - P(H|e_n)$. This gives us a definite iterative procedure based on well-accepted Bayesian belief networks for updating our policy on trust[Pea88]. It can easily be seen that eqn. (B.5) has this form, but lacks a methodology for rational policy-making.

The advantage of a Bayesian interpretation of policy then, is that it fits well with the notion of trust as a policy decision.

Appendix C
Proofs

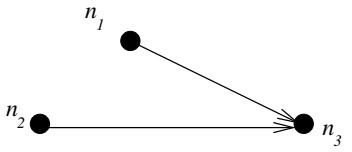


Fig. C.1. Mutual coordination by third-party mediation.

Lemma 3 (Observably consistent promises) *Two promises made to a third-party node, n_3, by two other nodes n_1 and n_2, are consistent (see fig. C.1), i.e.*

$$n_1 \xrightarrow{b} n_3 \quad \Leftrightarrow \quad n_2 \xrightarrow{b} n_3, \tag{C.1}$$

if and only if the two nodes additionally promise one another cooperation $C(b)$ in the matter of b.

Proof: Consistency implies

$$n_1 \xrightarrow{b} n_3 \quad \Leftrightarrow \quad n_2 \xrightarrow{b} n_3, \tag{C.2}$$

but from, eqn. 8.13 we may write

$$n_1 \xrightarrow{b} n_3 \Leftarrow n_1 \xrightarrow{C(b)} n_2 \otimes n_2 \xrightarrow{b} n_3 \tag{C.3}$$

$$n_2 \xrightarrow{b} n_3 \Leftarrow n_2 \xrightarrow{C(b)} n_1 \otimes n_1 \xrightarrow{b} n_3 \ . \tag{C.4}$$

Thus, using eqn. (C.2) on both sides of these equations, we derive

$$n_2 \overset{C(b)}{\to} n_1 \quad \Leftrightarrow \quad n_1 \overset{C(b)}{\to} n_2 \ . \tag{C.5}$$

■

Moreover, since the collaborative promise implies the existence of a promise of type b to a common third party, the reverse direction is also true. By measuring p from both n_1 and n_2, n_3 acts as a judge of their compliance with the mutual agreements between them (see fig. 6.2). This allows the basis of a theory of measurement or observation in collaborative networks.

References

[AB02] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

[Abb92] L.F. Abbott. *Acta Physica Polonica*, B13:33, 1992.

[ABH] D. Aredo, M. Burgess, and S. Hagen. A promise theory view on the policies of object orientation and the service oriented architecture. In *submitted to Science of Computer Programming*.

[ADL05] L. Arlotti, A. Deutsch, and M. Lachowicz. On a discrete boltzmann type model of swarming. *Math. Comp. Model*, 41:1193–1201, 2005.

[AR97] A. Abdul-Rahman. The pgp trust model. *EDI-Forum: the Journal of Electronic Commerce*, 1997.

[Ati81] P.S. Atiyah. *Promises, Morals and Law*. Clarendon Press, Oxford, 1981.

[Axe84] R. Axelrod. *The Evolution of Co-operation*. Penguin Books, 1990 (1984).

[Axe97] R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*. Princeton Studies in Complexity, Princeton, 1997.

[BA99] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[Bal97] V.K. Balakrishnan. *Graph Theory*. Schaum's Outline Series (McGraw-Hill), New York, 1997.

[Bal04] P. Ball. *Critical Mass, How one thing leads to another*. Random House, 2004.

[Bar02] A.L. Barabási. *Linked*. (Perseus, Cambridge, Massachusetts), 2002.

[BB08] J. Bergstra and M. Burgess. A static theory of promises. Technical report, arXiv:0810.3294v1, 2008.

[BBJF] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes. Summary of the stability of service level agreements. In *Proceedings of International Policy Workshop 2005.*

[BBK94] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS), LNCS*, volume 875, pages 3–18. Springer, 1994.

[BCE04a] M. Burgess, G. Canright, and K. Engø. A graph theoretical model of computer security: from file access to social engineering. *International Journal of Information Security*, 3:70–85, 2004.

[BCE04b] M. Burgess, G. Canright, and K. Engø. Inportance-ranking functions from the eigenvectors of directed graphs. *Journal of the ACM (Submitted)*, 2004.

[BdL13] Jan A. Bergstra and Karl de Leeuw. Bitcoin and beyond: Exclusively informational money. `http://arxiv.org/abs/1304.4758v2 [cs.CY]`, 2013.

[BDT99] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford, 1999.

[Ber01] C. Berge. *The Theory of Graphs*. (Dover, New York), 2001.

[BFa] M. Burgess and S. Fagernes. Pervasive computing management: A model of network policy with local autonomy. *IEEE Transactions on Software Engineering*, page (submitted).

[BFb] M. Burgess and S. Fagernes. Voluntary economic cooperation in policy based management. *IEEE Transactions on Network and Service Management*, page (submitted).

[BF06] M. Burgess and S. Fagernes. Autonomic pervasive computing: A smart mall scenario using promise theory. *Proceedings of the 1st IEEE International Workshop on Modelling Autonomic Communications Environments (MACE); Multicon verlag 2006. ISBN 3-930736-05-5*, pages 133–160, 2006.

[BF07] M. Burgess and S. Fagernes. Norms and swarms. *Lecture Notes on Computer Science*, 4543 (Proceedings of the first International Conference on Autonomous Infrastructure and Security (AIMS)):107–118, 2007.

[BG02] E. Brousseau and J-M. Glachant, editors. *The Economics of Contracts Theory and Applications*. Cambridge University Press, 2002.

[Bis02] M. Bishop. *Computer Security: Art and Science*. Addison Wesley, New York, 2002.

[BLMR03] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo. Using event calculus to formalise policy specification and analysis. In *Proceedings of the 4th IEEE Workshop on Policies for Distributed Systems and Networks*, 2003.

[BLMR04] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo. A goal-based approach to policy refinement. In *Proceedings of the 5th IEEE Workshop on Policies for Distributed Systems and Networks*, 2004.

[Bon87] P. Bonacich. Power and centrality: a family of measures. *American Journal of Sociology*, 92:1170–1182, 1987.

[BP06] J.A. Bergstra and A. Ponse. Interface groups for analystical execution architecture. *(preprint)*, 2006.

[Bura] M. Burgess. Business alignment through the eye-glass of promises. In *Keynote to BDIM workshop and NOMS2008, Brasil*.

[Burb] M. Burgess. Promise you a rose garden. http://research.iu.hio.no/papers/rosegarden.pdf.

[Bur93] M. Burgess. Cfengine www site. *http://www.cfengine.org*, 1993.

[Bur95] M. Burgess. A site configuration engine. *Computing systems (MIT Press: Cambridge MA)*, 8:309, 1995.

[Bur00] M. Burgess. Evaluation of cfengine's immunity model of system maintenance. *Proceedings of the 2nd international system administration and networking conference (SANE2000)*, 2000.

[Bur02] M. Burgess. *Classical Covariant Fields*. Cambridge University Press, Cambridge, 2002.

[Bur03] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.

[Bur04a] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.

[Bur04b] M. Burgess. Configurable immunity model of evolving configuration management. *Science of Computer Programming*, 51:197, 2004.

[Bur05] Mark Burgess. An approach to understanding policy based on autonomy and voluntary cooperation. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775*, pages 97–108, 2005.

[Bur12] M. Burgess. *New Research on Knowledge Management Models and Methods.*, chapter What's wrong with knowledge management? The emergence of ontology. Number ISBN 979-953-307-226-4. InTech, 2012.

[Bur13] M. Burgess. *In Search of Certainty: the science of our information infrastructure*. Xtaxis Press, 2013.

[Car84] J.P.W. Cartwright. An evidentiary theory of promises. *Mind (New Series)*, 93(370):230–248, 1984.

[CD] J.D. Carrillo and M. Dewatripont. Promises, promises. Technical Report 172782000000000058, UCLA Department of Economics, Levines's Bibliography.

[CD98] G. Di Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[CD01] A. Couch and N. Daniels. The maelstrom: Network service debugging via "ineffective procedures". *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 63, 2001.

[CEM04] G. Canright and K. Engø-Monsen. A natural definition of clusters and roles in undirected graphs. *Science of Computer Programming*, 53:195, 2004.

[Che80] B.F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. (J.Wiley & Sons., New York), 1991.

[DDLS00] N. Damianou, N. Dulay, E.C. Lupu, and M. Sloman. Ponder: a language for specifying security and management policies for distributed systems. *Imperial College Research Report DoC 2000/1*, 2000.

[DKS02] D. Daly, G. Kar, and W. Sanders. Modeling of service-level agreements for composed services. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 4, 2002.

[DMS99] L. Dondeti, S. Mukherjee, and A. Samal. Survey and comparison of secure group communication protocols, 1999.

[DS97] A. Daskalopulu and M. Sergot. The representation of legal contracts. *AI & Society*, 11:6–17, 1997.

[Dun96] R. Dunbar. *Grooming, Gossip and the Evolution of Language*. Faber and Faber, London, 1996.

[ea] J. Sauvé et al. Sla design from a business perspective. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775*.

[ea05a] J.M. Hendrickx et al. Rigidity and persistence of three and higher dimensional forms. In *Proceedings of the MARS 2005 Workshop on Multi-Agent Robotic Systems*, page 39, 2005.

[ea05b] J.M. Hendrickx et al. Structural persistence of three dimensional autonomous formations. In *Proceedings of the MARS 2005 Workshop on Multi-Agent Robotic Systems*, page 47, 2005.

[ea05c] X. Zhao et. al. Toward a formal theory of belief, capability, and promise incorporating temporal aspect. *LNAI (CEEMAS 2005)*, 3690:296–305, 2005.

[FB01] D. Fahrenholtz and A. Bartelt. Towards a sociological view of trust in computer science. In *Proceedings of the Eighth Research Symposium on Emerging Electronic Markets (RSEEM 01)*, page 10, 2001.

[FB04] S. Fagernes and M. Burgess. The effects of 'tit for tat' policy for rejecting 'spam' or denial of service floods. In *Proceedings of the 4th System Administration and Network Engineering Conference (SANE 2004)*, 2004.

[Fer99] J. Ferber. *Multi-agent Systems: Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.

[FL10] Jose M. Framinam and Rainer Leisten. Available-to-promise (atp) systems: a classification and framework for analysis. *Int. Journal of Production Research*, 48(11):3079–3103, 2010.

[Fox81] M.S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-1:70–80, 1981.

[Fri81] C. Fried. *Contract as promises*. Harvard University Press, 1981.

[FW01] Z. Fu and S.F. Wu. Automatic generation of ipsec/vpn security policies in an intra-domain environment. *Proceedings of the 12th internation workshop on Distributed System Operation and Management (IFIP/IEEE).*, INRIA Press:279, 2001.

[Gao09] W. Gao. Process management and orchestration. Master's thesis, Oslo University and Oslo University College, 2009.

[Gil93] M. Gilbert. Is an agreement and exchange of promises? *Journal of Philosophy*, 90(12):627–649, 1993.

[GMP92] J. Glasgow, G. MacEwan, and P. Panagaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10:226–264, 1992.

[GS01] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes (3rd edition)*. Oxford scientific publications, Oxford, 2001.

[Hey07] F. Heylighen. *Open Source Jahrbuch*, chapter Why is Open Access Development so Successful? Stigmergic organization and the economics of information. Lehrmanns Media, 2007.

[HJS04] T. Dong Huynh, Nicholas R. Jennings, and Nigel R. Shadbolt. Developing an integrated trust and reputation model for open multi-agent systems. In Rino Falcone, Suzanne Barber, Jordi Sabater, and Munindar Singh, editors, *AAMAS-04 Workshop on Trust in Agent Societies*, 2004.

[Hol98] J.H. Holland. *Emergence: from chaos to order*. Oxford University Press, 1998.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. Internet x.509 public key infrastructure: Certificate and certificate revocation list (crl) profile. http://tools.ietf.org/html/rfc3280, 2002.

[HR94] A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. J. Wiley & Sons, New York, 1994.

[Hui00] C. Huitema. *Routing in the Internet (2nd edition)*. Prentice Hall, 2000.

[Hum78] D. Hume. *Treatise on Human Nature*. Oxford, 1978.

[IT93] ITU-T. *Open Systems Interconnection - The Directory: Overview of Concepts, models and service. Recommendation X.500.* International Telecommunications Union, Geneva, 1993.

[JKD05] Audun Jøsang, Claudia Keser, and Theo Dimitrakos. Can we manage trust? In *Proceedings of the Third International Conference on Trust Management (iTrust), Versailes*, 2005.

[JP05] Audun Jøsang and Simon Pope. Semantic constraints for trust transitivity. In *APCCM '05: Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling*, pages 59–68, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[KE01] J. Kennedy and R.C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann (Academic Press), 2001.

[Kri63] S.A. Kripke. Semantical considerations in modal logic. *Acta Philosophica Fenica*, 16:83–94, 1963.

[KW05] J. Klüwer and A. Waaler. Trustworthiness by default, 2005.

[KW06] J. Klüwer and A. Waaler. Relative trustworthiness. In *Formal Aspects in Security and Trust: Third International Workshop, FAST 2005, Newcastle upon Tyne, UK, July 18-19, 2005, Revised Selected Papers, Springer Lecture Notes in Computer Science 3866*, pages 158–170, 2006.

[LaP94] L. LaPadula. A rule-set approach to formal modelling of a trusted computer system. *Computing systems (University of California Press: Berkeley, CA)*, **7**:113, 1994.

[LM05] A.L. Lafuente and U. Montanari. Quantitative mu-calculus and ctl defined over constraint semirings. *Electronic Notes on Theoretical Computing Systems QAPL*, pages 1–30, 2005.

[LP97] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation, Second edition*. Prentice Hall, New York, 1997.

[LS96] E.C. Lupu and M. Sloman. Towards a role based framework for distributed systems management. *Journal of Network and Systems Management*, **5**, 1996.

[LS97] E. Lupu and M. Sloman. Conflict analysis for management policies. In *Proceedings of the Vth International Symposium on Integrated Network Management IM'97*, pages 1–14. Chapman & Hall, May 1997.

[LST] A. Lim, V. Srinivasan, and C-K Tham. A comparative study of cooperative algorithms for wireless ad hoc networks. In *Unpublished*, pages 1–8.

[McC03] W.D. McComb. *Renormalization Methods: A Guide for Beginners*. Oxford University Press, 2003.

[McI89] M.D. McIlroy. Virology 101. *Computing systems (University of California Press: Berkeley, CA)*, **2**:173, 1989.

[Min05] Naftaly Minsky. Law governed interaction (lgi): A distributed coordination and control mechanism (an introduction, and a reference manual). Technical report, Rutgers University, June 2005.

[MP00] N. Minksy and P-P. Pal. Law-governed interaction: A coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology*, 9:273–305, 2000.

[MR72] J. Marschak and R. Radner. *Economic Theory of Teams*. Yale University Press, 1972.

[Mye91] R.B. Myerson. *Game theory: Analysis of Conflict*. (Harvard University Press, Cambridge, MA), 1991.

[Nas96] J.F. Nash. *Essays on Game Theory*. Edward Elgar, Cheltenham, 1996.

[NM44] J.V. Neumann and O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, Princeton, 1944.

[Nor01a] W.B. Norton. The art of peering: The peering playbook. Technical report, Equinix.com, 2001.

[Nor01b] W.B. Norton. Internet service providers and peering. Technical report, Equinix.com, 2001.

[NSW01] M. E. J. Newman, S.H. Strogatz, and D.J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64:026118, 2001.

[Ort98] R. Ortalo. A flexible method for information system security policy specifications. *Lecture Notes on Computer Science*, 1485:67–85, 1998.

[Ost90] E. Ostrom. *Governing the Commons*. Cambridge, 1990.

[Par01] J. Parrow. *An Introduction to the π-Calculus, in The Handbook of Process Algebra*, page 479. Elsevier, Amsterdam, 2001.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgen Kaufmann, San Francisco, 1988.

[Pep09] S. Pepper. *Encyclopedia of Library and Information Sciences*, chapter Topic Maps. CRC Press, ISBN 9780849397127, 2009.

[Pia95] J. Piaget. *Sociological Studies*. Routlege, London, 1995.

[PJ04] M. Patton and A. Jøsang. Technologies for trust in electronic commerce. *Electronic Commerce Research Journal*, 4:9–21, 2004.

[PS97] H. Prakken and M. Sergot. Dyadic deontic logic and contrary-to-duty obligations. In *Defeasible Deontic logic: Essays in Nonmonotonic Normative Reasoning*, volume 263 of *Synthese library*. Kluwer Academic Publisher, 1997.

[Qui98] A. Quinton. *Hume*. Pheonix, 1998.

[Rap70] A. Rapoport. *N-Person Game Theory: Concepts and Applications*. Dover, New York, 1970.

[Ras83] J. Rasmussen. Skills, rules, and knowledge; signals, signs and symbols, and other distinctions in humans performance models. *IEEE Transactions on Systems, Man and Cybernetics*, 13:257, 1983.

[Ras01] E. Rasmusen. *Games and Information (Third edition)*. Blackwell publishing, Oxford, 2001.

[Rec97] ITU-T Recommendation. X.509 (1997 e): Information technology - open systems interconnection - the directory: Authentication framework. Technical report, 1997.

[Rei65] F. Reif. *Fundamentals of statistical mechanics*. McGraw-Hill, Singapore, 1965.

[Rod02] G.B. Rodosek. Quality aspects in it service management. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 82, 2002.

[S00] Kazadi S. *Swarm Engineering*. PhD thesis, California Institute of Technology, 2000.

[SAB+01] R. Sailer, A. Acharya, M. Beigi, R. Jennings, and D. Verma. Ipsecvalidate - a tool to validate ipsec configurations. *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 19, 2001.

[SB04] M. Sallé and C. Bartolini. Management by contact. In *Proceedings of 8th Network Operations and Management Symposium NOMS 2004*. IEEE/IFIP, 2004.

[SBMR07] A. Sriraman, K.R.B. Butler, P.D. McDaniel, and P. Raghavan. Analysis of the ipv4 address space delegation structure. *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pages 501–508, July 2007.

[Sca90] T. Scanlon. Promises and practices. *Philosophy and Public Affairs*, 19(3):199–226, 1990.

[sch60] *The Strategy of Conflict*. Harvard Univesity Press, Cambridge, Mass., 1960.

[She11a] H. Sheinman. *Promies and Agreements*, chapter Introduction: promises and agreements, pages 3–57. Oxford University Press, 2011.

[She11b] H. Sheinman, editor. *Promises and Agreements*. Oxford, 2011.

[SM93] M.S. Sloman and J. Moffet. Policy hierarchies for distributed systems management. *Journal of Network and System Management*, 11(9):1404, 1993.

[Sny81] L. Snyder. Formal models of capability-based protection systems. *IEEE Transactions on Computers*, 30:172, 1981.

[Sto52] S.J. Stoljar. The ambiguity of promise. *Northwestern University law Review*, 47(1):1–20, 1952.

[Str07] J. Strassner. *Handbook of Network and System Administration*, chapter Knowledge Engineering Using Ontologies. Elsevier Handbook, 2007.

[SW49] C.E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1949.

[Wat99] D.J. Watts. *Small Worlds*. (Princeton University Press, Princeton), 1999.

[Wes01] D.B. West. *Introduction to Graph Theory (2nd Edition)*. (Prenctice Hall, Upper Saddle River), 2001.

[Win96] I.S. Winkler. The non-technical threat to computing systems. *Computing systems (MIT Press: Cambridge MA)*, **9**:3, 1996.

[Woo02] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, Chichester, 2002.

[WS03] Feng Wan and Munindar P. Singh. Commitments and causality for multiagent design. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2003.

[ZM03] S. Zhou and R.J. Mondragon. Analyzing and modelling the AS-level Internet topology. *ArXiv Computer Science e-prints*, March 2003.