Introduction to Choreographies
First edition (DRAFT)

Fabrizio Montesi
Department of Mathematics and Computer Science
University of Southern Denmark

Last update: September 3, 2019

Contents

Preface: Alice, Bob, Concurrency, and Distribution

We live in the era of *concurrency*, the performance of multiple tasks at a time, and *distribution*, the act of computing using communicating connected devices. These aspects have become pervasive in modern computing. Today, even mobile phones and tiny computers like Raspberry Pis feature multiple processing units of different kinds, with purposes that go from generic computing to more specialised ones like graphics and artificial intelligence. Our computer networks are getting bigger than ever, with the rise of the World Wide Web, telecommunications, cloud computing, and the Internet of Things. This transformation is making the number of computer programs that communicate with each other over a network explode. By 2025, the Internet alone is expected to connect from 25 to 100 billion devices [OECD 2016].

On the one hand, modern computer networks and their applications have become the drivers of our technological advancement. They give us better citizen services, a more efficient industry (Industry 4.0), new ways to connect socially, and even better health with smart medical devices. On the other hand, these systems and their software are increasingly complex. Services depend on other services to function. For example, a web store might depend on an external service provided by a bank to carry out customer payments. The web store, the customer's web browser, and the bank service are thus integrated: they communicate with each other to reach the goal of transferring the right amount of money to the right recipient, such that the customer can get the product she wants from the store. In concurrent and distributed systems, the heart of integration is the notion of *protocol*: a document that prescribes the communications that the involved parties should perform in order to reach a goal.

It is important that protocols are clear and precise. If they are ambiguous, we risk that the designers of different parts of the same system interpret them differently, which leads to errors. The consequences of such errors can be dire, including data getting in the system (deadlocks), data corruption, or data leaks. The more we equip programmers with solid methods for specifying and implementing protocols correctly, the more likely they are to succeed at integrating the different parts of concurrent and distributed systems correctly. The ultimate quest is to increase the reliability of these systems.

It is also important that protocols are as concise as possible: the bigger a protocol or the harder it is to write it down, the higher the chance that we make some mistake in writing it. Computer scientists and mathematicians might get a familiar feeling when presented with the necessity of achieving clarity, precision, and conciseness in writing. A computer scientist could point out that we need a good *language* to write protocols. A mathematician could say that we need a good *notation*. There is no contradiction here, since a language is, in fact, a notation. Both computer scientists and mathematician might say that we could (or should!) be even more ambitious and seek an *elegant* language.

Needham and Schroeder introduced an interesting notation for writing protocols in 1978. A communication of a message M from the participant A to the participant B is written

$$A \text{ -> } B : M \,.$$

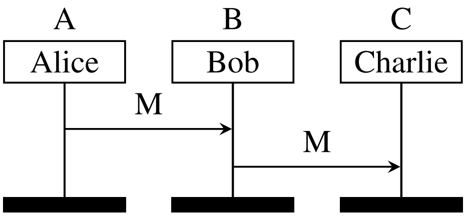
To define a protocol where A sends a message M to B, and then B passes the same message to C, we can just compose communications in a sequence:

$$\begin{array}{l} A \text{ -> } B : M \\ B \text{ -> } C : M \,. \end{array}$$

This notation is called “Alice and Bob notation”, due to a presentational style found in security research whereby the participants A and B represent the fictional characters Alice and Bob, respectively, who use the protocol to perform some task. There might be more participants, like C in our example—typically a shorthand for Carol, or Charlie. The first mention of Alice and Bob appeared in the seminal paper by Rivest et al. [1978] on their public-key cryptosystem:

> “For our scenarios we suppose that A and B (also known as Alice and Bob) are two users of a public-key cryptosystem”.

Over the years, researchers and developers created many more protocol notations. Some of these notations are graphical rather than textual, like Message Sequence Charts [International Telecommunication Union 1996]. The message sequence chart of our protocol with Alice, Bob, and Charlie looks as follows.



For our particular example, the graphical representation of our protocol (as a message sequence chart) and our previous textual representation (in Alice and Bob notation) are equivalent, in the sense that they contain the same information. This is not always the case: not all notations are equally expressive.

In the beginning of the 2000s, researchers and practitioners took the idea of protocol notations even further, and developed the idea of *choreography*. A choreography offers more details. For example, some details that might be included are:

- the kind of data being transmitted, or even the actual functions used to compute the data to be transmitted;
- nested protocols, i.e., the ability to call another protocol like a procedure;
- the state of participants, e.g., their memory states.

Choreographies are typically written in languages designed to be readable *mechanically*. This makes them amenable to be used in computer programs and, also, rigorous mathematical reasoning. In this book, we will start with a very simple choreography language and then progressively extend it with more sophisticated features, like parallelism and recursion. We will see that it is possible to define mathematically a *semantics* for choreographies, which gives us an interpretation of what running a protocol means. We will also see that it is possible to translate choreographies to a theoretical model of executable programs, which gives us an interpretation of how choreographies can be correctly implemented in the real world.

Although choreographies still represent a young and active area of research, they have already emerged in many places. In 2005, the World Wide Web Consortium (W3C)—the main international standards organisation for the web—drafted the Web Services Choreography Description Language (WS-CDL), for defining interactions among web services [W3C WS-CDL Working Group 2004]. In 2011, the global technology standards consortium Object Management Group (OMG) introduced choreographies in their notation for business processes [Object Management Group 2011]. The recent paradigm of microservices [Dragoni et al.

2017] advocates the use of choreographies to achieve better scalability. All this momentum is motivating a lot of research on both the theory of choreographies and its application to programming [Ancona et al. 2016, Hüttel et al. 2016]. These days, Alice and Bob surely are in the spotlight.

This book

This book is an introduction to the basic theory of choreographies. It explains what choreographies are and how we can model them mathematically. Its primary intended audience consists of computer science students and researchers, but it is also designed to be approachable by mathematicians (willing to become) familiar with context-free grammars.

The aim of this book is to be pedagogical, and to equip the reader with a new perspective on how we can abstract, design, and reason about concurrent and distributed systems. It is not an aim to be comprehensive, and we will not present features to capture all possible protocols. References to alternative notations, further developments, and techniques to model choreographies are given where appropriate. It is assumed that the reader is familiar with the notion of concurrency and the basic intuition of how distributed systems are programmed.

Prerequisites To read this book, you should be familiar with:

- discrete mathematics and the induction proof method;
- context-free grammars (only basic knowledge is required);
- basic data structures, like trees and graphs;
- concurrent and distributed systems.

These prerequisites are attainable in most computer science B.Sc. degrees, or with three years of relevant experience.

To study choreographies, we are going to define choreography languages and then write choreographies as terms of these languages. The syntax of languages is going to be defined using context-free grammars. To give meaning to choreographies, we are going to use extensively Plotkin's structural approach to operational semantics [Plotkin 2004].

The rules defining the semantics of choreographies are going to be rules of inference, borrowing from deductive systems. Knowing formal systems based on rules of inference is an advantage, but not a requirement for reading this book:

chapter 1 provides a brief introduction to the essential knowledge on these systems that we need for the rest of the book. The reader familiar with inference systems or structural operational semantics can safely skip the first chapter and jump straight to **??**.

An important aspect of choreographies is determining how they can be executed correctly in concurrent and distributed systems, in terms of independent programs for *processes*. To model process programs, we will borrow techniques from the area of process calculi. We will introduce the necessary notions on process calculi as we go along, so knowing this area is not a requirement for reading this book. The reader familiar with process calculi will recognise that we borrow many ideas from Milner's seminal calculus of communicating systems [Milner 1980].

Some exercises in this book are marked with !, indicating a higher degree of difficulty. For the exercises marked with $\hookrightarrow$, a solution is given in appendix A.

Chapter 1

Inference systems

Before we venture into the study of choreographies, we need to become familiar with the formalism that we are going to use throughout this book: inference systems. Inference systems are widely used in formal logic and the specification of programming languages (our case).[1]

Definition 1 (Inference systems and inference rules)**.** *An inference system is a set of* inference rules *(also called rules of inference). An inference rule has the form*

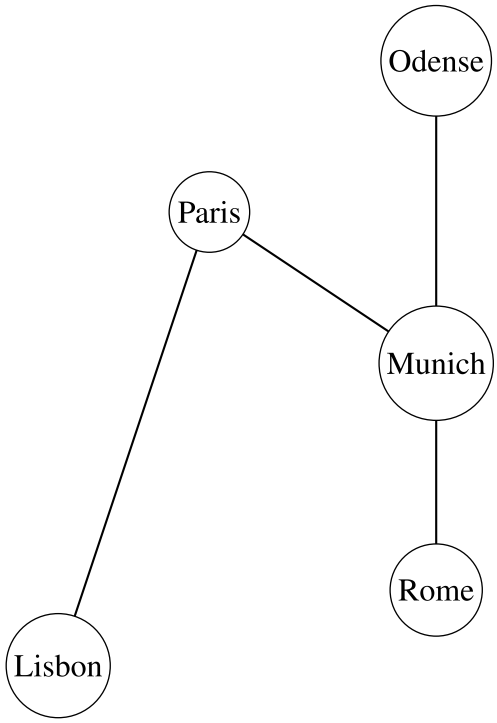
$$\frac{\textit{Premise } 1 \quad \textit{Premise } 2 \quad \cdots \quad \textit{Premise } n}{\textit{Conclusion}}$$

and reads "If the premises Premise 1*, Premise* 2*,* $\ldots$*, and Premise* n *hold, then Conclusion holds". It is perfectly valid for an inference rule to have no premises: we call such rules* axioms*, because their conclusions always hold. An inference rule has always exactly one conclusion.*

1.1 Example: Flight connections

Consider the following undirected graph of direct flights between cities.

[1] For further reading on these systems, see the lecture notes by Martin-Löf [1996].



Let A, B, and C range over cities. We denote that two cities A and B are connected by a direct flight with the *proposition* $\mathsf{conn}(A, B)$. Then, we can represent our graph as the set of axioms below.[2]

$$\frac{}{\mathsf{conn}(\text{Odense}, \text{Munich})} \qquad \frac{}{\mathsf{conn}(\text{Munich}, \text{Rome})} \qquad \frac{}{\mathsf{conn}(\text{Paris}, \text{Munich})}$$

$$\frac{}{\mathsf{conn}(\text{Paris}, \text{Lisbon})}$$

Notice that our graph is undirected, meaning that all direct flights are available also in the opposite direction. This is not faithfully represented by our axioms: if $\mathsf{conn}(A, B)$, it should also be the case that $\mathsf{conn}(B, A)$. One option to solve this discrepancy is to double the number of our axioms, to include their symmetric versions—e.g., we would add an axiom concluding with $\mathsf{conn}(\text{Munich}, \text{Odense})$. A more concise option is to formalise the general concept of symmetry of connections with a rule of inference, as follows.

$$\frac{\mathsf{conn}(A, B)}{\mathsf{conn}(B, A)}\ \textsc{Sym}$$

The label SYM is the name of our new inference rule; it is just a decoration to remember what the rule does (SYM is a shorthand for symmetry). Rule SYM tells

[2] This example is inspired by Pfenning's lecture notes on inference rules, where he also uses inference rules to model graphs [Pfenning 2012].

$$\frac{}{\mathsf{conn}(\text{Odense}, \text{Munich})} \quad \frac{}{\mathsf{conn}(\text{Munich}, \text{Rome})} \quad \frac{}{\mathsf{conn}(\text{Paris}, \text{Munich})}$$

$$\frac{}{\mathsf{conn}(\text{Paris}, \text{Lisbon})}$$

$$\frac{\mathsf{conn}(A, B)}{\mathsf{conn}(B, A)}\ \textsc{Sym} \qquad \frac{\mathsf{conn}(A, B)}{\mathsf{path}(A, B)}\ \textsc{Dir} \qquad \frac{\mathsf{path}(A, B) \quad \mathsf{path}(B, C)}{\mathsf{path}(A, C)}\ \textsc{Trans}$$

Figure 1.1: An inference system for flights.

us that if we have a connection from any A to any B (premise), then we have a connection from B to A (conclusion). In this rule, A and B are *schematic variables*: they can stand for any of our cities. So if we were to add more connections in the future, their symmetric version would also become immediately available thanks to rule SYM.

Now that we are satisfied with how our inference system captures the graph, we can use it to find flight paths from a city to another. Say that for any two cities A and B, the proposition $\mathsf{path}(A, B)$ denotes that there is a path from A to B. Finding paths is relatively easy. First, we observe that direct connections give us a path. (In the following rule, DIR stands for direct.)

$$\frac{\mathsf{conn}(A, B)}{\mathsf{path}(A, B)}\ \textsc{Dir}$$

Second, we formulate a rule for multi-step paths. If there is a path from A to B, and a path from B to C, then we have a path from A to C. In other words, paths are transitive. (TRANS stands for transitivity.)

$$\frac{\mathsf{path}(A, B) \quad \mathsf{path}(B, C)}{\mathsf{path}(A, C)}\ \textsc{Trans}$$

The whole system is displayed in fig. 1.1.

1.2 Derivations

The key feature of an inference system is the capability of performing *derivations*. Suppose that we wanted to answer the following question.

> Is there a flight path from Odense to Rome?

Answering this question in our inference system for flight connections corresponds to showing that $\mathsf{path}(\text{Odense}, \text{Rome})$ holds. To do this, we have to build a *derivation tree*. We will call derivation trees also simply *derivations*, or *proof trees* (since they prove that something can be derived in a given inference system).

From Odense to Rome We start looking for our derivation from what we want to conclude with—our conclusion is what we want to prove.

$$\mathsf{path}(\text{Odense}, \text{Rome})$$

Observe that we have only two inference rules that can conclude something of this form: Dir and Trans. Let us try to apply Dir first, by substituting its schematic variables with the cities that we need:

$$\frac{\mathsf{conn}(\text{Rome}, \text{Odense})}{\mathsf{path}(\text{Odense}, \text{Rome})}\ \textsc{Dir}\ .$$

Our derivation is not complete, because we are left with a premise that we are now responsible for proving: $\mathsf{conn}(\text{Odense}, \text{Rome})$. We can read what we have now as: if Rome is connected to Odense, then there is a path from Odense to Rome. However, proving $\mathsf{conn}(\text{Odense}, \text{Rome})$ seems unfeasible: there is no rule that concludes that, and we cannot hope to conclude it by applying rule Sym, since that would require in turn to prove $\mathsf{conn}(\text{Rome}, \text{Odense})$.

Now that we know that rule Dir is not helpful for our derivation, our only remaining option is to apply rule Trans. Clearly, we should instantiate A as Odense and C as Rome in the application of Trans. How should we instantiate B, though? (We informally write $B?$ for "we do not know what B should be yet" here.)

$$\frac{\mathsf{path}(\text{Odense}, B?) \quad \mathsf{path}(B?, \text{Rome})}{\mathsf{path}(\text{Odense}, \text{Rome})}\ \textsc{Trans}$$

One way would be to try out all possible cities as our B, in search of a city that will allow us to continue our derivation. However, looking at the definition of our graph, it is easy to see that the right choice is to pick Munich, since that city is connected to both Odense and Rome.

$$\frac{\mathsf{path}(\text{Odense}, \text{Munich}) \quad \mathsf{path}(\text{Munich}, \text{Rome})}{\mathsf{path}(\text{Odense}, \text{Rome})}\ \textsc{Trans}$$

What we have above is an instantiation of rule Trans. (We will call rule instantiations also rule applications in the remainder, since it corresponds to "applying" the rule to the premises in order to reach the conclusion.) The derivation depends on the validity of some premises. We can read it as a proof of the statement “If $\mathsf{path}(\text{Odense}, \text{Munich})$ holds and $\mathsf{path}(\text{Munich}, \text{Rome})$ holds, then $\mathsf{path}(\text{Odense}, \text{Rome})$ holds”. In other words, our derivation of $\mathsf{path}(\text{Odense}, \text{Rome})$ has two hypotheses: $\mathsf{path}(\text{Odense}, \text{Munich})$ and $\mathsf{path}(\text{Munich}, \text{Rome})$. Thus, we have gone from the task of deriving $\mathsf{path}(\text{Odense}, \text{Rome})$ to the tasks of deriving $\mathsf{path}(\text{Odense}, \text{Munich})$ and $\mathsf{path}(\text{Munich}, \text{Rome})$, respectively. Thanks to the notation of inference rules, we can complete these tasks just by “digging deeper” with further rule applications. Since we know that Odense and Munich are connected, we can try using rule Dir.

$$\dfrac{\dfrac{\mathsf{conn}(\text{Odense}, \text{Munich})}{\mathsf{path}(\text{Odense}, \text{Munich})}\,\textsc{Dir} \qquad \mathsf{path}(\text{Munich}, \text{Rome})}{\mathsf{path}(\text{Odense}, \text{Rome})}\,\textsc{Trans}$$

We now have to prove $\mathsf{conn}(\text{Odense}, \text{Munich})$. We have that as the conclusion of one of our axioms, so we can conclude that branch of our derivation by applying the related axiom.

$$\dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\text{Odense}, \text{Munich})}}{\mathsf{path}(\text{Odense}, \text{Munich})}\,\textsc{Dir} \qquad \mathsf{path}(\text{Munich}, \text{Rome})}{\mathsf{path}(\text{Odense}, \text{Rome})}\,\textsc{Trans}$$

We can finish our derivation for the right premise $\mathsf{path}(\text{Munich}, \text{Rome})$ likewise.

$$\dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\text{Odense}, \text{Munich})}}{\mathsf{path}(\text{Odense}, \text{Munich})}\,\textsc{Dir} \qquad \dfrac{\dfrac{}{\mathsf{conn}(\text{Munich}, \text{Rome})}}{\mathsf{path}(\text{Munich}, \text{Rome})}\,\textsc{Dir}}{\mathsf{path}(\text{Odense}, \text{Rome})}\,\textsc{Trans}$$

What we have now is a *complete* derivation, i.e., a derivation without hypotheses. We can tell by the fact that there are no more premises left to prove. Since the derivation is complete, it shows that $\mathsf{path}(\text{Odense}, \text{Rome})$ can always be derived in our inference system, without making any assumptions. In general, we say that a proposition is derivable if there exists at least one complete derivation with such proposition as conclusion.

The structure of derivations We now move to the formal definition of derivations.

First, some notation. Let $\mathcal{D}$ (for derivation) range over derivations and p range over propositions. (Sometimes we will also write $\mathcal{P}$, for “proof”, to denote a derivation.) We shall write $\overset{\mathcal{D}}{p}$ for "the derivation $\mathcal{D}$ has conclusion p". For example, assume that $\mathcal{D}$ is our latest derivation, as follows.

$$\mathcal{D} \quad \triangleq \quad \cfrac{\cfrac{\overline{\mathsf{conn}(\text{Odense}, \text{Munich})}}{\mathsf{path}(\text{Odense}, \text{Munich})}\ \text{\textsc{Dir}} \qquad \cfrac{\overline{\mathsf{conn}(\text{Munich}, \text{Rome})}}{\mathsf{path}(\text{Munich}, \text{Rome})}\ \text{\textsc{Dir}}}{\mathsf{path}(\text{Odense}, \text{Rome})}\ \text{\textsc{Trans}}$$

The symbol $\triangleq$ in the equation above means "is defined as". Therefore, we have that $\overset{\mathcal{D}}{\mathsf{path}(\text{Odense}, \text{Rome})}$.

Observe that $\mathcal{D}$ can be seen as a tree, by viewing each rule application as a node of the tree, where the root node is the rule application that reaches the conclusion of the derivation. (In the last derivation, the root is the application of rule Trans.) This is a very useful property that we will use throughout the book, in particular to apply the principle of induction in our formal reasonings about derivations.

Definition 2 (Derivation)**.** *Let S be an inference system. Then:*

- *Any application of a rule in S is a derivation in S.*
- *Let $\mathcal{D}_1$, …, $\mathcal{D}_n$ be derivations in S, for some natural number $n > 0$, with respective conclusions p_1, …, p_n. Then, any application of a rule* R *in S with p_1, …, p_n as premises is a derivation in S.*

Intuitively, a complete derivation is a derivation where the "leaves" of the derivation tree (the base cases) are all axioms. We can formalise this by tuning slightly the base case of the previous definition. (The difference is emphasised.)

Definition 3 (Complete Derivation)**.** *Let S be an inference system.*

- *Any application of* an axiom *in S is a complete derivation in S.*
- *Let $\mathcal{D}_1$, …, $\mathcal{D}_n$ be complete derivations in S, for some natural number $n > 0$, with respective conclusions p_1, …, p_n. Then, any application of a rule* R *in S with p_1, …, p_n as premises is a complete derivation in S.*

We impose that derivations are finite: we are interested in proofs that can be checked mechanically in finite time.

Exercise 1 ($\hookrightarrow$)**.** *Consider the system in fig. 1.2, which replaces rules* Dir *and* Trans *respectively with the alternative rules* DirW *and* TransW*, which measure the length of a path (paths are "weighted", with each connection having weight* 1*).*

Prove that, for any A and B, if $\mathsf{path}(A, B)$ *is derivable in the system in fig. 1.1, then there exists n such that* $\mathsf{path}(A, B, n)$ *is derivable in the system in fig. 1.2.*

Suggestion: proceed by structural induction on the proof of $\mathsf{path}(A, B)$*.*

$$\frac{}{\mathsf{conn}(\text{Odense}, \text{Munich})} \quad \frac{}{\mathsf{conn}(\text{Munich}, \text{Rome})} \quad \frac{}{\mathsf{conn}(\text{Paris}, \text{Munich})}$$

$$\frac{}{\mathsf{conn}(\text{Paris}, \text{Lisbon})}$$

$$\frac{\mathsf{conn}(A, B)}{\mathsf{conn}(B, A)}\ \textsc{Sym}$$

$$\frac{\mathsf{conn}(A, B)}{\mathsf{path}(A, B, 1)}\ \textsc{DirW} \qquad \frac{\mathsf{path}(A, B, n) \quad \mathsf{path}(B, C, m)}{\mathsf{path}(A, C, n+m)}\ \textsc{TransW}$$

Figure 1.2: Weighted rules for flight paths.

1.3 Underivable propositions

Showing that a proposition holds requires showing a proof, i.e., a derivation, in the inference systems of interest. Once the proof is shown, then we just need to check that all rules have been applied correctly. If we are convinced that this is the case, then we are convinced that the proof is correct and that the proposition indeed holds.

Knowing what can be derived is paramount to establish the adequacy of an inference system, but it is just as important to check what *cannot* be derived. But how can we prove that something cannot be derived? This can be tricky, because it requires us to reason about all the possible derivations that could, potentially, conclude with our proposition and showing that none of those derivations can actually be built.

Consider a simple example: showing that $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ is not derivable. Intuitively, there is no direct connection between Lisbon and Rome in our graph. But how can we show this formally?

First, we observe that the only rules that can have a conclusion of the form $\mathsf{conn}(A, B)$ for any A and B are our axioms and rule Sym. This observation conveniently restricts the set of rules that we have to consider. None of the axioms can be applied for $A = \text{Lisbon}$ and $B = \text{Rome}$, as in our case. We are left only with rule Sym, so any search of a derivation of $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ must begin as follows.

$$\frac{\mathsf{conn}(\text{Rome}, \text{Lisbon})}{\mathsf{conn}(\text{Lisbon}, \text{Rome})}\ \textsc{Sym}$$

Again, by the same argument, there is no rule to apply for $\mathsf{conn}(\text{Rome}, \text{Lisbon})$ but Sym. So we obtain that the proof *must* continue with another application of

$$\frac{}{\mathsf{conn}(\text{Odense}, \text{Munich})} \qquad \frac{}{\mathsf{conn}(\text{Munich}, \text{Rome})} \qquad \frac{}{\mathsf{conn}(\text{Paris}, \text{Lisbon})}$$

$$\frac{\mathsf{conn}(A, B)}{\mathsf{conn}(B, A)}\ \textsc{Sym}$$

$$\frac{\mathsf{conn}(A, B)}{\mathsf{path}(A, B, 1)}\ \textsc{DirW} \qquad \frac{\mathsf{path}(A, B, n) \quad \mathsf{path}(B, C, m)}{\mathsf{path}(A, C, n + m)}\ \textsc{TransW}$$

Figure 1.3: A limited and weighted flight system.

Sym.

$$\cfrac{\cfrac{\mathsf{conn}(\text{Lisbon}, \text{Rome})}{\mathsf{conn}(\text{Rome}, \text{Lisbon})}\ \textsc{Sym}}{\mathsf{conn}(\text{Lisbon}, \text{Rome})}\ \textsc{Sym}$$

We got back to where we started: we have to prove $\mathsf{conn}(\text{Lisbon}, \text{Rome})$. Since we have only one way to prove it, and we have just shown that it leads to the exact same premise, this points out that our proof search will go on indefinitely and that we will never reach a finite derivation. So, $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ cannot be proven.

Let us be more formal, to convince ourselves that $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ cannot be derived more decisively. Since every derivation is a finite tree, we can measure the height of a derivation as a natural number (it is the height of the tree). Thus, among all the proofs of $\mathsf{conn}(\text{Lisbon}, \text{Rome})$, there is at least one of minimal height, in the sense that there is no other proof of $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ with lower height. We attempt at finding this minimal proof. The reasoning goes as before, and we soon end up seeing that a minimal proof necessarily starts as follows.

$$\cfrac{\cfrac{\mathsf{conn}(\text{Lisbon}, \text{Rome})}{\mathsf{conn}(\text{Rome}, \text{Lisbon})}\ \textsc{Sym}}{\mathsf{conn}(\text{Lisbon}, \text{Rome})}\ \textsc{Sym}$$

So we have to find some proof of our premise $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ on top. But if such a proof exists, it would be a proof of $\mathsf{conn}(\text{Lisbon}, \text{Rome})$ that is *smaller* than the proof that we are building (because it would not have the first two applications of Sym of our proof). Thus our minimal proof must be bigger than another proof, and we reach a contradiction.

Exercise 2. *Consider the system in fig. 1.3, which removes the direct flight from Paris to Munich.*

Prove that it is not possible to derive $\mathsf{path}(\mathit{Lisbon}, \mathit{Munich}, 1)$.

$$\frac{}{\mathsf{conn}(\text{Odense}, \text{Munich})} \quad \frac{}{\mathsf{conn}(\text{Munich}, \text{Rome})} \quad \frac{}{\mathsf{conn}(\text{Paris}, \text{Munich})}$$

$$\frac{}{\mathsf{conn}(\text{Paris}, \text{Lisbon})}$$

$$\frac{\mathsf{conn}(A, B)}{\mathsf{conn}(B, A)}\ \textsc{Sym} \qquad \frac{\mathsf{conn}(A, B)}{\mathsf{path}(A, B)}\ \textsc{Dir} \qquad \frac{\mathsf{conn}(A, B) \quad \mathsf{path}(B, C)}{\mathsf{path}(A, C)}\ \textsc{Step}$$

Figure 1.4: An alternative way of constructing paths.

Exercise 3. *Prove that it is not possible to derive* $\mathsf{path}(\mathit{Lisbon}, \mathit{Munich}, 2)$ *using the system in fig. 1.3.*

Exercise 4 (!)**.** *Prove that there exists no* n *such that* $\mathsf{path}(\mathit{Lisbon}, \mathit{Munich}, n)$ *is derivable in the system in fig. 1.3.*

1.4 Rule derivability and admissibility

Consider the system in fig. 1.4, which replaces rule TRANS from fig. 1.1 with rule STEP. The difference is that rule STEP requires a direct connection from the source A to some city B, and then a path from B to the destination C.

From an algorithmic perspective, adopting rule TRANS or rule STEP might lead to slightly different search strategies. Searching for a path from A to C using rule STEP roughly corresponds to: look up in our database of direct connections (given by the axioms and their reflexive closure, thanks to rule SYM) from A to some B, and then recursively try to find a path from B to C; if we fail, we have to try with another B, if possible. By contrast, searching for a path from A to C using rule TRANS corresponds to recursively trying to find a path from A to some B, and then again recursively trying to find a path from B to C; again, if we fail, we have to try with another B, if possible. Of course, these strategies are only for paths not covered already by rule DIR, which covers the case in which A and C are connected directly.

Since the two systems are different, a key question is whether they are *equally powerful*, in the sense that every derivable proposition in one of the two is derivable also in the other.

There are only two kinds of propositions that we can derive in our two systems: $\mathsf{conn}(A, B)$ and $\mathsf{path}(A, B)$. It is easy to see that, for any A and B, $\mathsf{conn}(A, B)$ is derivable in the system with rule TRANS if and only if it is derivable in the system with rule STEP, simply because the two systems share exactly the same rules for deriving conn propositions. For propositions of the form path(A, B), the situation is more complicated. We tackle the two directions separately (from the system with rule STEP to the system with rule TRANS, and vice versa). The exploration of each direction leads to its own useful new concept—derivable rules and admissible rules.

1.4.1 Derivable rules

We start by showing that the system with rule TRANS (fig. 1.1) can derive all paths that can be derived in the system with rule STEP (fig. 1.4).

To do this, we prove that adding rule STEP to the system with rule TRANS would not add any new derivable propositions. Recall that rule STEP looks as follows.

$$\frac{\mathsf{conn}(A, B) \qquad \mathsf{path}(B, C)}{\mathsf{path}(A, C)}\ \textsc{Step}$$

The key observation here is that we can build a derivation that, from the premises conn(A, B) and path(B, C), concludes path(A, C) by using the rules in fig. 1.1. Here it is.

$$\frac{\dfrac{\mathsf{conn}(A, B)}{\mathsf{path}(A, B)}\ \textsc{Dir} \qquad \mathsf{path}(B, C)}{\mathsf{path}(A, C)}\ \textsc{Trans} \tag{1.1}$$

The derivation above holds for any A, B, and C. Since it is parametric on these schematic variables, a more correct name might be derivation scheme, but we will allow ourselves the abuse of terminology and simply call it a derivation. This derivation is proof that rule STEP is *derivable* in the system in fig. 1.1. In general, we say that a rule is derivable whenever its conclusion can be derived from its premises by using rules that are already in the system. In other words, if we can build a derivation from the premises of the rule to its conclusion, then the rule is derivable.

A very nice and convenient property of derivable rules is that their applications can be rewritten locally inside of derivations, without the need for modifying the derivations of the premises. Let us see an example to understand what this means. From the derivation in eq. (1.1), we now know that we can rewrite every application of rule STEP into a valid derivation in the system in fig. 1.1 as follows. We also show where the derivations of the premises go ($\rightarrow$ here means "is rewritten to”):

$$\frac{\overset{\mathcal{D}}{\mathsf{conn}(A,B)} \qquad \overset{\mathcal{E}}{\mathsf{path}(B,C)}}{\mathsf{path}(A,C)}\ \textsc{Step} \quad \to \quad \frac{\dfrac{\overset{\mathcal{D}}{\mathsf{conn}(A,B)}}{\mathsf{path}(A,B)}\ \textsc{Dir} \qquad \overset{\mathcal{E}}{\mathsf{path}(B,C)}}{\mathsf{path}(A,C)}\ \textsc{Trans} \quad . \tag{1.2}$$

We can apply this transformation to any derivation in the system with rule Step (fig. 1.4). For instance, consider the following derivation of a multi-hop path from Odense to Lisbon:

$$\frac{\dfrac{}{\mathsf{conn}(\text{Odense},\text{Munich})} \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\text{Paris},\text{Munich})}}{\mathsf{conn}(\text{Munich},\text{Paris})}\ \textsc{Sym} \qquad \dfrac{\dfrac{}{\mathsf{conn}(\text{Paris},\text{Lisbon})}}{\mathsf{path}(\text{Paris},\text{Lisbon})}\ \textsc{Dir}}{\mathsf{path}(\text{Munich},\text{Lisbon})}\ \textsc{Step}}{\mathsf{path}(\text{Odense},\text{Lisbon})}\ \textsc{Step}$$

To translate this derivation into a derivation in the system in fig. 1.1, we can just rewrite each application of rule Step as indicated by eq. (1.2). The order in which we pick and rewrite these applications does not matter.[3] For example, here is the result of replacing the top-right occurrence of Step first:

$$\frac{\dfrac{}{\mathsf{conn}(\text{Odense},\text{Munich})} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\text{Paris},\text{Munich})}}{\mathsf{conn}(\text{Munich},\text{Paris})}\ \textsc{Sym}}{\mathsf{path}(\text{Munich},\text{Paris})}\ \textsc{Dir} \qquad \dfrac{\dfrac{}{\mathsf{conn}(\text{Paris},\text{Lisbon})}}{\mathsf{path}(\text{Paris},\text{Lisbon})}\ \textsc{Dir}}{\mathsf{path}(\text{Munich},\text{Lisbon})}\ \textsc{Trans}}{\mathsf{path}(\text{Odense},\text{Lisbon})}\ \textsc{Step} \quad .$$

And here the result of replacing also the reamining occurrence:

$$\frac{\dfrac{\dfrac{}{\mathsf{conn}(\text{Odense},\text{Munich})}}{\mathsf{path}(\text{Odense},\text{Munich})}\ \textsc{Dir} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\text{Paris},\text{Munich})}}{\mathsf{conn}(\text{Munich},\text{Paris})}\ \textsc{Sym}}{\mathsf{path}(\text{Munich},\text{Paris})}\ \textsc{Dir} \qquad \dfrac{\dfrac{}{\mathsf{conn}(\text{Paris},\text{Lisbon})}}{\mathsf{path}(\text{Paris},\text{Lisbon})}\ \textsc{Dir}}{\mathsf{path}(\text{Munich},\text{Lisbon})}\ \textsc{Trans}}{\mathsf{path}(\text{Odense},\text{Lisbon})}\ \textsc{Trans} \quad . \tag{1.3}$$

What we have in eq. (1.3) is a complete derivation that is valid in the system in fig. 1.1.

[3] In fact, thanks to the locality of our transformation, one could even devise a concurrent algorithm that replaces all occurrences of Step in parallel.

1.4.2 Rule admissibility

We now move to the other direction: proving that adding rule TRANS to the system with rule STEP would not add any new derivable propositions.

Recall that rule TRANS is defined as follows.

$$\dfrac{\mathsf{path}(A,B) \qquad \mathsf{path}(B,C)}{\mathsf{path}(A,C)}\ \textsc{Trans}$$

As a first attempt, we could try the same strategy that we followed in section 1.4.1: deriving the conclusion $\mathsf{path}(A,C)$ from the premises $\mathsf{path}(A,B)$ and $\mathsf{path}(B,C)$ using the rules in fig. 1.4.

Unfortunately, we reach a dead end pretty quickly when trying to show that rule TRANS is derivable in the system with rule STEP. The only way to build a path with multiple connections is by using rule STEP, which requires a conn as premise. But our only available premises are path propositions, and we have no rule that allows us to conclude conn from a path.

We resort to a different proof technique and show that rule TRANS is *admissible* in the system with rule STEP (fig. 1.4). An admissible rule is one that does not add any new derivable propositions. All derivable rules are also admissible, but admissible rules are not necessarily derivable (just like our case here with rule TRANS). It is sometimes convenient to mark admissible rules and their applications explicitly. Here, we will distinguish them by using a dashed horizontal line.

Theorem 1. *The rule*

$$\begin{array}{c}\mathsf{path}(A,B) \qquad \mathsf{path}(B,C) \\ \hdashline \mathsf{path}(A,C)\end{array}\ \textsc{Trans}$$

is admissible in the system in fig. 1.4.

Proof. We need to prove that, for every derivation $\mathcal{D}$ using the rules in fig. 1.4 and rule TRANS, there exists a derivation with the same conclusion that uses only the rules in fig. 1.4. To shorten our text, we shall say that a derivation is TRANS-free when it does not contain any applications of rule TRANS.

We proceed by induction on the size of $\mathcal{D}$. We have a case for each one of the rules that can be applied last in $\mathcal{D}$. For the induction to work, we prove a stronger result: for every derivation $\mathcal{D}$ using the rules in fig. 1.4 and rule TRANS, there exists a derivation with the same conclusion that uses only the rules in fig. 1.4 *and is not bigger than* $\mathcal{D}$. This last remark about size is going to be important in the last case of this proof.

Case $\mathcal{D}$ ends with one of the axioms. In these cases $\mathcal{D}$ is trivially TRANS-free (as it consists only of the axiom), so the thesis holds by picking $\mathcal{D}$.

Case $\mathcal{D}$ ends with an application of rule SYM:

$$\mathcal{D} \;=\; \frac{\overset{\mathcal{D}'}{\mathsf{conn}(A,B)}}{\mathsf{conn}(B,A)}\;\textsc{Sym}$$

for some derivation $\mathcal{D}'$.

All rules that can conclude $\mathsf{conn}(A,B)$ are either axioms or rules that have only conn propositions as premises. Since TRANS does not conclude with a conn proposition, $\mathcal{D}'$ is necessarily TRANS-free and so is $\mathcal{D}$.

Case $\mathcal{D}$ ends with an application of rule DIR:

$$\mathcal{D} \;=\; \frac{\overset{\mathcal{D}'}{\mathsf{conn}(A,B)}}{\mathsf{path}(A,B)}\;\textsc{Dir}$$

for some derivation $\mathcal{D}'$.

By following the same reasoning for the previous case, we know that $\mathcal{D}'$ and $\mathcal{D}$ are TRANS-free.

Case $\mathcal{D}$ ends with an application of rule TRANS:

$$\mathcal{D} \;=\; \frac{\overset{\mathcal{E}}{\mathsf{path}(A,B)} \quad \overset{\mathcal{F}}{\mathsf{path}(B,C)}}{\mathsf{path}(A,C)}\;\textsc{Trans}$$

for some derivations $\mathcal{E}$ and $\mathcal{F}$.

This is the most interesting case. By induction hypothesis, we know that there exists a TRANS-free derivation $\mathcal{E}'$ such that $\overset{\mathcal{E}'}{\mathsf{path}(A,B)}$.

There are only two possibilities for how $\mathcal{E}'$ ends: either with an application of rule DIR or with an application of rule STEP. This gives us two subcases.

Case For some $\mathcal{E}''$,

$$\mathcal{E}' \;=\; \frac{\overset{\mathcal{E}''}{\mathsf{conn}(A,B)}}{\mathsf{path}(A,B)}\;\textsc{Dir}\quad .$$

The thesis follows from the derivation

$$\frac{\overset{\mathcal{E}''}{\mathsf{conn}(A,B)} \quad \overset{\mathcal{F}'}{\mathsf{path}(B,C)}}{\mathsf{path}(A,C)}\;\textsc{Step}\quad .$$

Case For some B', $\mathcal{E}_1$ and $\mathcal{E}_2$,

$$\mathcal{E}' = \frac{\overset{\mathcal{E}'_1}{\mathsf{conn}(A,B')} \quad \overset{\mathcal{E}'_2}{\mathsf{path}(B',B)}}{\mathsf{path}(A,B)}\ \textsc{Step} \quad .$$

Consider the following derivation $\mathcal{G}$.

$$\mathcal{G} \triangleq \frac{\overset{\mathcal{E}'_2}{\mathsf{path}(B',B)} \quad \overset{\mathcal{F}}{\mathsf{path}(B,C)}}{\mathsf{path}(B',C)}\ \textsc{Trans}$$

The derivation $\mathcal{G}$ is smaller than $\mathcal{D}$: $\mathcal{E}'_2$ is part of $\mathcal{E}'$, which by induction hypothesis is not bigger than $\mathcal{E}$; and $\mathcal{F}$ is part of $\mathcal{D}$. Thus, we can invoke the induction hypothesis on $\mathcal{G}$ and get that there exists a Trans-free derivation $\mathcal{G}'$ with the same conclusion: $\overset{\mathcal{G}'}{\mathsf{path}(B',C)}$.

The thesis now follows by:

$$\frac{\overset{\mathcal{E}'_1}{\mathsf{conn}(A,B')} \quad \overset{\mathcal{G}'}{\mathsf{path}(B',C)}}{\mathsf{path}(A,C)}\ \textsc{Step} \quad .$$

□

Example 1. *To the reader not familiar with this kind of proofs, the last case might look like a bit of a magic trick! How are we manipulating the derivation, exactly?*

Let us look at a concrete, nontrivial example of a proof $\mathcal{D}$ containing applications of Trans *(to help readability, we use the same derivation names as in the proof of theorem 1):*

$$\mathcal{D} \triangleq \frac{\overset{\mathcal{E}}{\mathsf{path}(\mathit{Odense},\mathit{Paris})} \quad \overset{\mathcal{F}}{\mathsf{path}(\mathit{Paris},\mathit{Lisbon})}}{\mathsf{path}(\mathit{Odense},\mathit{Lisbon})}\ \textsc{Trans}$$

where

$$\mathcal{E} \triangleq \frac{\dfrac{\dfrac{}{\mathsf{conn}(\mathit{Odense},\mathit{Munich})}}{\mathsf{path}(\mathit{Odense},\mathit{Munich})}\ \textsc{Dir} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\mathit{Paris},\mathit{Munich})}}{\mathsf{conn}(\mathit{Munich},\mathit{Paris})}\ \textsc{Sym}}{\mathsf{path}(\mathit{Munich},\mathit{Paris})}\ \textsc{Dir}}{\mathsf{path}(\mathit{Odense},\mathit{Paris})}\ \textsc{Trans} \quad .$$

$$\mathcal{F} \triangleq \frac{\dfrac{}{\mathsf{conn}(\mathit{Paris},\mathit{Lisbon})}}{\mathsf{path}(\mathit{Paris},\mathit{Lisbon})}\ \textsc{Dir}$$

We now "run" the proof of theorem 1 on $\mathcal{D}$. We are clearly in the last case, so we start by obtaining a TRANS*-free derivation $\mathcal{E}'$ that has the same conclusion as $\mathcal{E}$:*

$$\mathcal{E}' \;=\; \dfrac{\dfrac{}{\mathsf{conn}(\mathit{Odense}, \mathit{Munich})} \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\mathit{Paris}, \mathit{Munich})}}{\mathsf{conn}(\mathit{Munich}, \mathit{Paris})}\,\textsc{Sym}}{\mathsf{path}(\mathit{Munich}, \mathit{Paris})}\,\textsc{Dir}}{\mathsf{path}(\mathit{Odense}, \mathit{Paris})}\,\textsc{Step} \;.$$

$\mathcal{E}'$ is not bigger than $\mathcal{E}$ (it is actually smaller), as expected. We have that

$$B' = \mathit{Munich}$$

$$\mathcal{E}'_1 = \dfrac{}{\mathsf{conn}(\mathit{Odense}, \mathit{Munich})}$$

$$\mathcal{E}'_2 = \dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\mathit{Paris}, \mathit{Munich})}}{\mathsf{conn}(\mathit{Munich}, \mathit{Paris})}\,\textsc{Sym}}{\mathsf{path}(\mathit{Munich}, \mathit{Paris})}\,\textsc{Dir}$$

$$\mathcal{G} = \dfrac{\overset{\mathcal{E}'_2}{\mathsf{path}(\mathit{Munich}, \mathit{Paris})} \qquad \overset{\mathcal{F}}{\mathsf{path}(\mathit{Paris}, \mathit{Lisbon})}}{\mathsf{path}(\mathit{Munich}, \mathit{Lisbon})}\,\textsc{Trans}$$

To obtain $\mathcal{G}'$, we apply recursively the same reasoning to $\mathcal{G}$, obtaining:

$$\mathcal{G}' \;=\; \dfrac{\dfrac{}{\mathsf{conn}(\mathit{Munich}, \mathit{Paris})} \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{conn}(\mathit{Paris}, \mathit{Munich})}}{\mathsf{conn}(\mathit{Munich}, \mathit{Paris})}\,\textsc{Sym}}{\mathsf{path}(\mathit{Munich}, \mathit{Paris})}\,\textsc{Dir}}{\mathsf{path}(\mathit{Munich}, \mathit{Lisbon})}\,\textsc{Step} \;.$$

The final TRANS*-free result is:*

$$\mathcal{D}' \;=\; \dfrac{\overset{\mathcal{E}'_1}{\mathsf{conn}(\mathit{Odense}, \mathit{Munich})} \qquad \overset{\mathcal{G}'}{\mathsf{path}(\mathit{Munich}, \mathit{Lisbon})}}{\mathsf{path}(\mathit{Odense}, \mathit{Lisbon})}\,\textsc{Step} \;.$$

Exercise 5 (!)**.** *Prove that if $\mathsf{path}(A, B)$ is derivable in the system in fig. 1.4, then there exists a nonempty sequence of derivable propositions $\mathsf{conn}(C_1, C'_1)$, …, $\mathsf{conn}(C_n, C'_n)$ for some C_1, …, C_n such that $n > 0$, $C_1 = A$ (the sequence starts from A), and $C'_n = B$ (the sequence ends at B).*

Prove that if $\mathsf{path}(A, B)$ is derivable in the system in fig. 1.1, then there exists a nonempty sequence of derivable propositions $\mathsf{conn}(C_1, C'_1)$, …, $\mathsf{conn}(C_n, C'_n)$ for some C_1, …, C_n such that $n > 0$, $C_1 = A$ (the sequence starts from A), and $C'_n = B$ (the sequence ends at B).

Appendix A

Solution to selected exercises

We give the solutions to some selected exercises, pointing out the main aspects. Some solutions are given in full detail, to serve as examples of exposition.

Solution of exercise 1. We prove only the direction from the system in fig. 1.1 to the system in fig. 1.2.

To prove this direction, we actually prove the stronger statement:

- if $\mathsf{conn}(A, B)$ is derivable in the system in fig. 1.1, then $\mathsf{conn}(A, B)$ is derivable in the system in fig. 1.2;
- if $\mathsf{path}(A, B)$ is derivable in the system in fig. 1.1, then there exists a natural number n such that $\mathsf{path}(A, B, n)$ is derivable in the system in fig. 1.2.

We proceed by induction on the structure of the derivation of $\mathsf{conn}(A, B)$ or $\mathsf{path}(A, B)$ in the system in fig. 1.1. We get a case for each rule that could be applied last in the derivation.

Base cases (axioms) If the last applied rule is an axiom, then the proof is valid also in the other system directly, since the two systems share the same axioms.

Case SYM The derivation has this shape:

$$\dfrac{\begin{array}{c}\mathcal{D}\\ \mathsf{conn}(B, A)\end{array}}{\mathsf{conn}(A, B)}\ \textsc{Sym}\ .$$

By induction hypothesis on the sub-derivation $\mathcal{D}$, we know that there exists $\mathcal{D}'$ in the other system such that:

$$\begin{array}{c}\mathcal{D}'\\ \mathsf{conn}(B, A)\end{array}.$$

The thesis follows by applying rule SYM:

$$\dfrac{\overset{\mathcal{D}'}{\mathsf{conn}(B,A)}}{\mathsf{conn}(A,B)}\ \text{SYM}\ .$$

Case DIR The derivation has this shape:

$$\dfrac{\overset{\mathcal{D}}{\mathsf{conn}(A,B)}}{\mathsf{path}(A,B)}\ \text{DIR}\ .$$

By induction hypothesis, we know that there exists $\mathcal{D}'$ in the other system such that:

$$\overset{\mathcal{D}'}{\mathsf{conn}(A,B)}\ .$$

The thesis follows by applying rule DIRW.

$$\dfrac{\overset{\mathcal{D}'}{\mathsf{conn}(A,B)}}{\mathsf{path}(A,B,1)}\ \text{DIRW}\ .$$

Case TRANS The derivation has this shape:

$$\dfrac{\overset{\mathcal{D}}{\mathsf{path}(A,B)} \quad \overset{\mathcal{E}}{\mathsf{path}(B,C)}}{\mathsf{path}(A,C)}\ \text{TRANS}\ .$$

By induction hypothesis on $\mathcal{D}$ and by induction hypothesis on $\mathcal{E}$, we know that there exist natural numbers n and m, and derivations $\mathcal{D}'$ and $\mathcal{E}'$ in the other system such that:

$$\overset{\mathcal{D}'}{\mathsf{path}(A,B,n)} \qquad \overset{\mathcal{E}'}{\mathsf{path}(A,B,m)}\ .$$

The thesis follows by applying rule TRANSW.

$$\dfrac{\overset{\mathcal{D}'}{\mathsf{path}(A,B,n)} \quad \overset{\mathcal{E}'}{\mathsf{path}(A,B,m)}}{\mathsf{path}(A,B,n+m)}\ \text{TRANSW}\ .$$

■

List of Figures

List of Notations

! A (possibly) difficult exercises. 10

$\hookrightarrow$ An exercise for which a solution is provided in appendix A. 10

Index

Bibliography

Davide Ancona, Viviana Bono, Mario Bravetti, Joana Campos, Giuseppe Castagna, Pierre-Malo Deniélou, Simon J. Gay, Nils Gesbert, Elena Giachino, Raymond Hu, Einar Broch Johnsen, Francisco Martins, Viviana Mascardi, Fabrizio Montesi, Rumyana Neykova, Nicholas Ng, Luca Padovani, Vasco T. Vasconcelos, and Nobuko Yoshida. Behavioral types in programming languages. *Foundations and Trends in Programming Languages*, 3(2-3):95–230, 2016.

Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering*, pages 195–216. Springer, 2017.

Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016.

International Telecommunication Union. Recommendation Z.120: Message sequence chart, 1996.

Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic journal of philosophical logic*, 1(1):11–60, 1996.

Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, Berlin, 1980.

R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, December 1978. ISSN 0001-0782. doi: 10.1145/359657.359659.

Object Management Group. Business Process Model and Notation. http://www.omg.org/spec/BPMN/2.0/, 2011.

OECD. Horizon scan of megatrends and technology trends in the context of future research policy, 2016. http://ufm.dk/en/publications/2016/an-oecd-horizon-scan-of-megatrends-and-technology-trends-in-the-context-of-future-research-policy.

F. Pfenning. Lecture Notes on Deductive Inference, 2012. https://www.cs.cmu.edu/ fp/courses/15816-s12/lectures/01-inference.pdf.

Gordon D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. ISSN 0001-0782. doi: 10.1145/359340.359342. URL http://doi.acm.org/10.1145/359340.359342.

W3C WS-CDL Working Group. Web services choreography description language version 1.0. http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/, 2004.