Using Scone's Multiple-Context Mechanism to Emulate Human-Like Reasoning

Scott E. Fahlman
Carnegie Mellon University, Language Technologies Institute
sef@cs.cmu.edu

Abstract

Scone is a knowledge-base system developed specifically to support human-like common-sense reasoning and the understanding of human language. One of the unusual features of Scone is its multiple-context system. Each context represents a distinct world-model, but a context can inherit most of the knowledge of another context, explicitly representing just the differences. We explore how this multiple-context mechanism can be used to emulate some aspects of human mental behavior that are difficult or impossible to emulate in other representational formalisms. These include reasoning about hypothetical or counter-factual situations; understanding how the world model changes over time due to specific actions or spontaneous changes; and reasoning about the knowledge and beliefs of other agents, and how their mental state may affect the actions of those agents.

The Scone Knowledge-Base System

Scone is a knowledge-representation and reasoning system – a *knowledge base* or KB – that has been developed over the last few years by the author's research group at Carnegie Mellon University (Fahlman 2006; see also www.cs.cmu.edu/~sef/scone/). Scone, by itself, is not a complete AI or decision-making system, and does not aspire to be; rather, it is a software component – a sort of smart active memory system – that is designed to be used in a wide range of software applications, both in AI and in other areas. Scone deals just with symbolic knowledge. Things like visualization, motor memory, and memory for sound sequences are also important for human-like AI, but we believe that those will have specialized representations of their own, linked in various ways to the symbolic memory.

Scone has been used in a number of applications at Carnegie Mellon and with a few selected outside partners; we plan a general open-source release of Scone in the near future, as soon as we can assemble the resources and infrastructure needed to support a larger external user community.

Scone occupies a different part of the design space from other KB systems currently in use – particularly systems such as OWL that are based on First-Order Logic or Description Logic. Our goal in developing Scone has been to support common-sense reasoning and natural-language understanding, not theorem-proving and logic puzzles. Therefore, we place primary emphasis on Scone's expressiveness, ease of use, and scalability.

For human-like common-sense reasoning, we need expressiveness that is *greater* than that of first-order logic, not a less-expressive subset of FOL. In particular, we need to use higher-order logical constructs and default reasoning with exceptions, as explained below. We also need a system that can scale up to millions of entities and statements, and perhaps tens or hundreds of millions, while still delivering something like real-time performance.

To achieve those goals simultaneously, we must give up the constraint of using only logically complete reasoning methods applied to a provably consistent knowledge base. Any system with the expressiveness we want can be proven to be intractable or undecidable. Logical proofs are a wonderful invention, but for a system with the goals of Scone, we can't afford to deal in proofs and completeness. Instead, Scone uses more limited and local forms of inference that appear to provide the power and accuracy we need for human-like reasoning. For a more thorough discussion of this trade-off, see (Fahlman 2008).

Like other knowledge base systems, Scone provides support for representing symbolic knowledge about the world: general common-sense knowledge, detailed knowledge about some specific application domain, or perhaps some of each. Scone also provides efficient support for simple inference: inheritance of properties in a type hierarchy, following chains of transitive relations, detection of type mismatches (e.g. trying to assert that a male person is someone's mother), and so on. Scone

supports flexible search within the knowledge base. For example, we can ask Scone to return all individuals or types in the KB that exhibit some set of properties, or we can ask for a best-match if no exact match is available.

It is not the purpose of this paper to provide a tutorial on Scone, but we will briefly describe some of Scone's more unusual features to provide the necessary background for our real topic: Scone's multiple-context mechanism and its uses.

Speed and scalability: Benchmark tests have shown that Scone can deliver real-time performance on a standard (~$1000) workstation, even as the KB grows to several million elements and statements. Scone's core search and inference capabilities (Fahlman, 2006) are based on marker-passing algorithms originally designed for a hypothetical massively parallel machine, the NETL machine (Fahlman, 1979). If we need more speed or a much larger KB, Scone's current algorithms can easily be mapped to certain kinds of parallel architectures.

Clean separation of conceptual vs. lexical or linguistic information: Scone may be viewed as a semantic network of nodes and links. The *nodes* in Scone represent concepts, not words or phrases. These nodes are generally written with curly braces, such as {elephant}, which represents the *concept* elephant, and is tied to other concepts by *relational links*. The names within the curly braces are arbitrary unique labels, but the concept {elephant} is tied to zero or more lexical items in English, and to additional lexical items in other languages. The concepts themselves are mostly universal and language independent – the concept {elephant} is more or less the same in every culture that knows about elephants – but Scone's multiple-context mechanism can be used to represent concepts that exist in one culture but not in another, or that are significantly different in some cultures.

Frame-like representation with virtual-copy semantics: In Scone and other frame-like KBs, there are arbitrarily complex structured descriptions (prototypes) for each type represented: elephant, family, contest, and so on. These concept descriptions typically have many interrelated parts, attributes, and relations to other individuals and types. Upon creating a subtype or instance, all of the structure of the prototype description is efficiently inherited. In effect, the new subtype or instance becomes a *virtual copy* of the parent description; it can then be specialized by adding new information or by subtracting information that would otherwise be inherited.

Default reasoning with exceptions: The virtual-copy mechanism described above is very powerful, but for real-world domains this inheritance would be nearly useless without the ability to cancel or over-ride some of the information that would otherwise be inherited. We want to say that a {bird} is a {flying thing} in general, but a {penguin} is a bird that can't fly. A {canary} is a normal bird – it is a {flying thing} – but some individual canaries cannot fly. These exceptions are not rare anomalies: the more you know about the prototypical properties of a given type, the less likely you are to ever find an instance that matches that prototype in every respect. The world is full of flightless birds, three-legged dogs, and even the occasional honest politician. But most current KB systems exclude any sort of cancellation (non-monotonic reasoning) because that can create all sorts of problems for a system that is trying to be logically complete. The ability to create exceptions and to explicitly over-ride some inherited information is a core capability of Scone, and is efficiently implemented in Scone's inference algorithms..

Higher-order logic and meta-information: First-order logic lacks the ability to make statements about statements, to reason about this meta-information, and to use that to control which statements will participate in the system's reasoning. In Scone, statements are treated like any other objects. We can attach attributes (meta-information) to these statements, and that can be used to selectively activate and de-activate statements. This is a form of higher-order logic, and it is used in Scone in a number of ways. For example, we might assign a confidence value and some source information to each statement in some domain. For some deductions, we might insist that only highly confident facts, or those from certain trusted sources, are to participate; the others are dormant. Scone's multiple-context system is essentially a convenient packaging of some of these higher-order capabilities.

Again, this is not a rare anomaly that can easily be ignored. Consider the following statement: "John believes that Fred knows Mary's phone number, but that's not really true." This is inherently a higher-order statement. And yet, without the ability to represent and reason about statements like this, we could not understand the plot of a typical sitcom, most of which involve misunderstandings and confusion, or (as we will see) the plot of a simple children's story that involves some type of deception.

Multiple Contexts in Scone

Perhaps the most unusual aspect of Scone is its *multiple-context* mechanism. In this section, I will describe what this is and how it works; in the remainder of the paper, I will describe how multiple contexts can be used to implement some aspects of human thought that are awkward or impossible for other knowledge-representation formalisms.

A *context* is simply a node in Scone's knowledge base that represents a distinct *world model* within the larger Scone KB. There can be any number of these contexts within the Scone system.

Every node N in Scone has a connection to some context-node C in which the entity represented by N exists. Every link L in Scone has a connection to a context node C within which the statement represented by L is valid. At any given time, one context node C_{ACTIVE} in the KB is said to be *active*, meaning that we are operating within the world-model that C_{ACTIVE} represents. The nodes and links within C_{ACTIVE} are available to participate in the system's reasoning; those not within C_{ACTIVE} are *dormant*, and do not participate in the system's reasoning. New information added to the Scone KB is generally placed in the current C_{ACTIVE}.

The Scone reasoning engine has efficient mechanisms to switch from one active context to another. The reasoning engine will often switch contexts many times while working on a given task. A useful rule of thumb is that the English word "in" usually (but not always) signals a context-switch. A context may represent a region of space ("*in* the UK…"); a region of time ("*in* the 1960s…"); an alternative reality ("*in* the fictional Harry Potter universe…"); a hypothetical state ("*in* the scenario where Sarah Palin is elected President…"); a mental state indicating the knowledge and beliefs of some person or other being ("*in* John's opinion…"); or a combination of these ("*in* Paris *in* the spring…"). There is a special large context {general} that holds – not surprisingly – things that are true "*in* general".

It would be possible, but terribly inefficient, to implement each context or world model as a separate KB, with its own private copy of all its knowledge. In most cases, one context will differ from another in only a few respects; the vast majority of their knowledge will be common to both of them. So in Scone, contexts are hierarchical: a context C_2 starts out as the clone of some existing parent context C_1, inheriting all of C_1's knowledge; then we can customize C_2 by adding some additional knowledge or by explicitly subtracting some knowledge that would otherwise be inherited. This inheritance is handled efficiently by Scone's marker-passing algorithms.

An example: Suppose we want to create new context representing the Harry Potter world. Call this C_{HP}. To create C_{HP}, we find some existing context that is close to what we want – in this case, "present day UK" or C_{UK}. We add the new C_{HP} node to the knowledge base, along with a "clone-of" link (a kind of "is-a" link) from C_{HP} to C_{UK}. Anything that exists in C_{UK} – houses, double-decker busses, London – now also exists in C_{HP}. Anything that we know to be true in C_{UK} is also true in C_{HP}: for example, people drive on the left side of the road. So far, C_{HP} is not very interesting, but now we can add some new entities – Harry himself, Hogwarts School, dragons – and some new facts – for example, "A broom is a vehicle". We can also explicitly cancel some things that would otherwise be inherited from C_{UK}, such as "The primary purpose of a broom is cleaning." If we activate the C_{HP} context, we see all this new information and can use it in reasoning, planning, or answering questions. If we derive any new information while working in C_{HP}, that is added to C_{HP}. If we activate C_{UK}, these changes are all invisible; the information in C_{HP} plays no part in any reasoning we might do in C_{UK}.

How is all this implemented in the Scone knowledge-base engine on a serial machine? Details can be found in (Fahlman 2006), but here is a very brief sketch: To activate C_{HP}, we put a special "active context" marker on the node representing C_{HP}. We then propagate this marker upwards across "is-a" and "clone-of" links to mark all the nodes from which C_{HP} is supposed to inherit information. In this case, the marker will end up on the CUK node, the {general} node, and probably many others. Our benchmark testing has shown that this context activation will typically take much less than a millisecond on a standard serial workstation.

In all of our subsequent reasoning, whenever we are about to make use of a node or link, we first check whether the context-node it is tied to is marked with the activation marker – that is, we check whether the context is active. If the mark is present, we proceed; if not, we ignore the presence of this node or link. That check adds a small, constant extra cost to each reasoning step, but once we have accepted that cost, we can have as many nested contexts as we like. The time required to switch contexts may go up as the number of contexts and their interconnections goes up, but there is no further slowdown of normal reasoning.

The multiple-context system used in Scone is a direct descendant (though with some changes) of the multiple-context system I proposed for the NETL knowledge-representation system over thirty years ago (Fahlman 1979). That, in turn was influenced by the Conniver language (Sussman and McDermott 1972). There have been many knowledge representation systems since then with multiple contexts or something similar, but I am not aware of any that are very similar to the system in Scone (or in NETL). For example, the micro-theories in Cyc (Lenat et al. 1990) are oriented more toward packaging the knowledge into internally consistent sub-domains, not all of which have to be loaded at once, than Scone's goal of easy movement between overlapping world-models (which may or may not be internally consistent).

Hypotheses and Counter-Factual Situations

It is the mark of an educated mind to be able to entertain a thought without accepting it. – Aristotle

An important survival skill for humans is the ability to represent and reason about counter-factual situations – that is, situations that differ in some respect from the current reality. *"What would I do if a hungry bear appeared at the mouth of my cave during the night? I could throw rocks at it. Maybe I should go collect some rocks now, while it's safe to do so…"* Even Aristotle, the inventor of formal logic, understood the importance of occasionally reasoning about situations that are not actually true in your current model of reality, though he seems to have believed that only educated people can do this – demonstrably untrue.

Scone's multiple-context mechanism makes it relatively straightforward to reason about counter-factuals and their consequences, without letting that exercise spill over into your current world-model. We begin by creating a new context, C*BEAR*, which is initially a clone of C*GENERAL*, inheriting all of its contents. We then customize C*BEAR*, which may involve adding new knowledge ("There's a hungry bear outside"), cancelling some inherited knowledge ("It's currently daytime"), or some of each.

These changes affect only C*BEAR*. They are not visible when C*GENERAL* is active. We can now do whatever reasoning or planning we want to do in C*BEAR*, resulting, perhaps, in the conclusion that (back in current reality), we should have a supply of rocks on hand. We can inject that goal back into C*GENERAL*, with a link to the C*BEAR* context as the explanation – a bear might appear in the night.

It should be obvious that the same approach can be used to entertain multiple, possibly inconsistent, hypotheses as we reason about the world or make plans: *"Perhaps these droppings at the mouth of the cave are from a bear – what would be the consequences? Or perhaps they are from a deer – different consequences altogether."*

It is possible to garbage-collect a hypothetical context, and all its contents, as soon as we are done with it, but it is also possible to keep any number of these contexts around indefinitely. Perhaps we have chosen one of N possibile interpretations of the mysterious droppings as most likely, but we want to retain the other possibilities and what we have deduced about their consequences, in case we get more evidence later.

Temporal States, Actions, and Events

Another essential human trait is our *episodic memory*: that is, our ability to represent and reason about events, actions, sequences of events or actions, plans, preconditions, and consequences. This is a very complex business, since it involves both representation and planning. (Since we are talking here about everyday human abilities, I will consider only "good enough" planning, not *optimal* planning.) Scone's multiple-context mechanism gives us a good way to handle the representational part of this problem.

In Scone, an *event* is just a description or frame with any number of type-restricted slots; an *action* is just an event with an event with a special slot for the *agent* – the entity that caused the event to take place (whatever that means – questions of causality get into deep philosophical waters pretty quickly, but people seem to have a usable informal model of causality sufficient for most everyday problems).

An event or action description has two additional important slots: the {before-context} and the {after-context}. These are two distinct world-models that represent the state of the world before the event and the state of the world after the event.

Consider the generic action {move}, in the sense of "move a physical object". The major slots in this description are the {agent}, the {object being moved}, {location-1}, {location-2}, and the {time of action}. For a given instance of {move} – call it {move-27} – all of these slots exist (directly or by inheritance), but we may or may not have fillers for all of them. For example, we may know who moved an object, but not when that occurred.

In the {before-context} of any {move}, the {location} of the {object being moved} is {location-1}. In the {after-context}, it is {location-2}. That's the essence of what {move} does. Both the {before-context} and the {after-context} are complete world-models. We can activate either of them and reason about what is true in that model. Perhaps location-2 is currently in the shade, so the object will cool down.

The {before-context} of {move-27} is initially a clone of the surrounding context in which the move occurs, so all of that general knowledge is inherited. The only thing we have to represent explicitly is the existence of the object and its location in this context. The {after-context} of {move-27} begins as a clone of the {before-context}. We just need to explicitly represent what has changed: the object's presence at location-1 is cancelled, and its presence at location-2 is added.

(In many ways, this model resembles the old STRIPS model of actions and planning (Fikes and Nilsson, 1971): the explicit contents of the {before-context} correspond to the pre-conditions of the action; the explicit contents of the {after-context} represent the post-conditions.)

Actions and events may be strung together to form a sequence (or, more generally, a lattice), with the {after-context} of one event being the {before-context} of the next one. An event may be thought of (and represented in Scone) as atomic – an indivisible change-of-state that transforms one state into another. Or it may have an {expansion} – a sequence of sub-events – that represent steps in the larger action.

The Frame Problem: What Has Changed?

In the description above, I said that the {after-context} of an action or event only has to represent explicitly what has changed relative to the {before-context}. But things are not really that simple. This touches upon some aspects of the famous Frame Problem (McCarthy and Hayes 1969). This paper highlighted some of the problems that can occur in the representation of world-states that change over time. There is now a very large literature on various aspects and manifestations of this problem.

Here we will look at three aspects of the Frame Problem that relate to this discussion. First, in some logical formalisms, it is difficult or impossible to say "World-model T2 is the same as world-model T1, except for some specific differences that we will state explicitly." To handle this well requires default reasoning with exceptions, which many logical formalisms are reluctant to include. As we have seen, Scone handles this in a very natural way.

Second, there is the question of how many changes to deduce and to describe explicitly. When I pick up a glass and move it to a new position, its location changes. We certainly want to represent that change. But I also may have left some fingerprints on the glass. We normally would never think about that, but in certain contexts (murder mysteries) that could be very significant. Also, that action perturbed *very slightly* the orbit of Jupiter. People who have studied classical physics know, or can deduce, that this is true, but it's hard to think of a context in which this would be important. This seems like an ideal candidate for *lazy evaluation*.

In Scone, we could note in the generic {move} description that these kinds of consequences are possible, but mark them as lazy. We would not work out the details in the specific {move-27} description until/unless we have some need for this information. Note that this marking is a form of higher-order logic, and the decision not to deduce all the consequences that we could deduce is a violation of logically complete reasoning, so many logic-based representations will have a serious problem with this.

Finally, there is the issue of things that change spontaneously, or at least due to some action/event that is not explicitly represented in our model. Ice melts (depending on the ambient temperature), food in the fridge goes bad (at a rate depending on the food and assorted other conditions). A cigarette butt dropped on the sidewalk will probably still be there the next day, but a $100 bill left lying on the sidewalk probably will not be. The so-called Yale Shooting Problem (Hanks and McDermott 1987; see also http://www-formal.stanford.edu/leora/commonsense/) is a good example of this phenomenon. Again, the question is how many of the almost infinite set of possible deductions should we perform pro-actively, and how do we handle the cases where we did not consider some change that becomes important later.

We have not yet attacked this genuinely hard problem using Scone. Clearly, once again, some sort of lazy evaluation is called for, and once again that will cause problems for those who insist on logically complete methods. One idea is to create phantom event-types for all the families of change that we might ever want to consider: ice "melts", food "spoils", and so on. Then when we are interested in the possible consequent of such an event, we can consider whether this phantom event is likely to have occurred. In any case, Scone will not complain about the logical incompleteness inherent in this style of reasoning.

Reasoning About Differing Mental States

We humans are able to model the *mental state* of other humans, and to reason about their plans and actions based on this model. For social animals like us, this is a very important capability.

Consider what is required to really understand – well enough to answer questions or make predictions – the following fragment of a children's story, easily understandable by an average human 4-year-old:

> *The last little pig is safe in his brick house. The wolf is outside, but the door is locked. The wolf can't get in as long as the door is locked. The pig wants to go outside, but he knows the wolf is waiting outside, so he will not unlock the door.*
>
> *The wolf decides to trick the pig. He stomps his feet on the ground hard, first one and then the other, and then stomps more and more softly, so it sounds like he has walked away. The pig falls for the trick and believes the wolf is gone. He unlocks the door to go out. The wolf has pork chops for dinner that night.*[1]

There are a number of interesting issues here, such as the qualitative-physics reasoning (Forbus 1988) required to understand why a series of ever-softer footstep noises would suggest that the stepper is walking away. But let's leave those for another day – the narrator explains the purpose of the stomping, so we don't have to figure that out. Instead, let's concentrate on the mental states at play here, and how we can represent actions that manipulate mental states such as "trick" (in the sense of "deceive").

In order to understand this story, it is necessary to represent the state of affairs in a number of different world models, without confusing the contents of one model with another. In Scone, each of these models is represented by a distinct context, as shown in Figure 1.

[1]This is a slight variation of the original Three Little Pigs story that I made up to emphasize the aspect of trickery or deception.

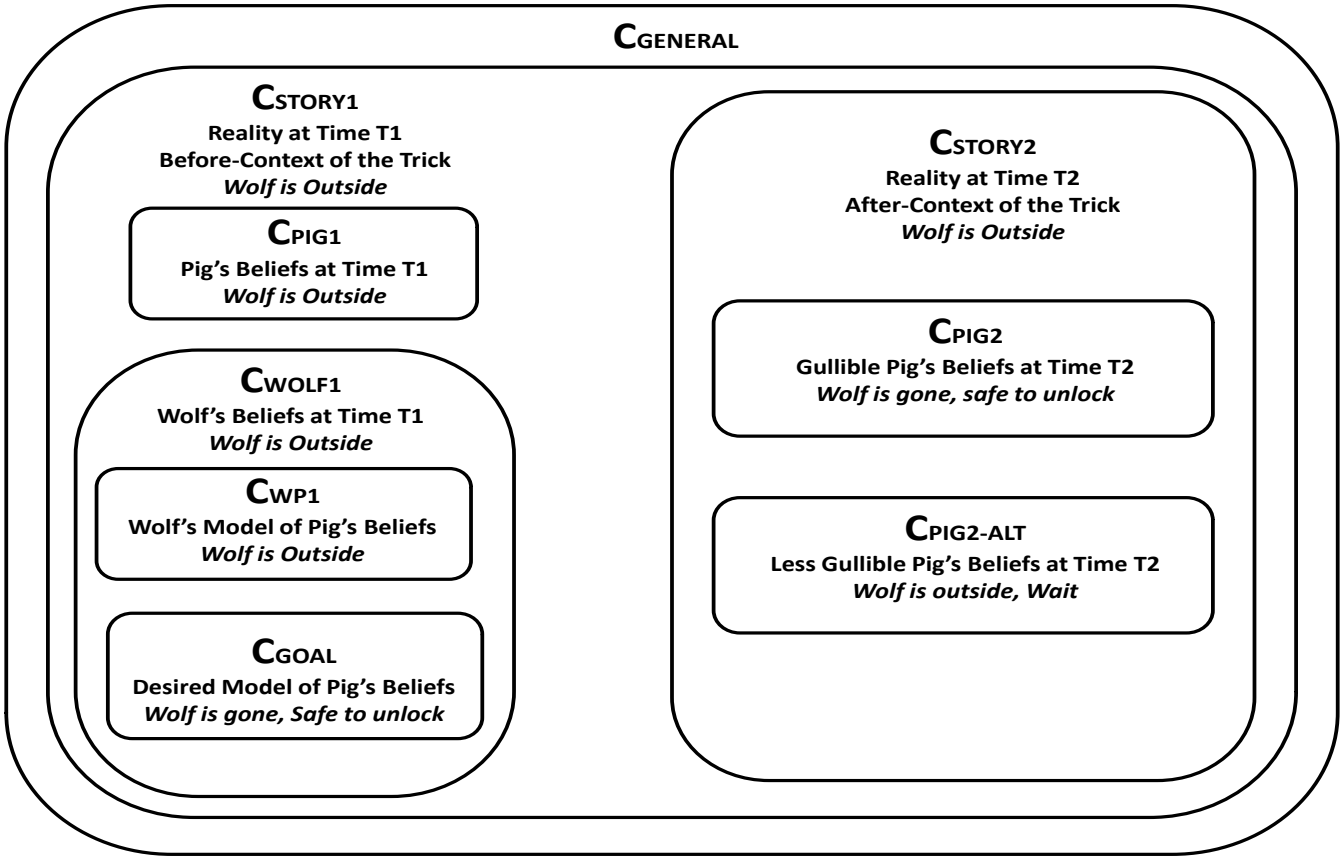

Figure 1: Use of multiple mental-state contexts in the wolf/pig story. Each context inherits by default the knowledge contained in the surrounding context, then can add or subtract some items.

At the beginning, we have the system's general knowledge context C*GENERAL*; the story-reality at this time (as described by the narrator) C*STORY1*; the pig's initial knowledge state C*PIG1*; and the wolf's initial knowledge state C*WOLF1*. We also might need a context representing the wolf's model of the pig's current mental state C*WP1*.

In C*GENERAL*, we have the system's current view of reality, with lots of general knowledge: brick houses are sturdy, you can't go through a locked door unless you unlock it or break it down, etc.

All of this general knowledge is inherited (by default) by C*STORY1*, which adds some new assertions and cancels some inherited ones to reflect a few key differences. First, this is a story, not a history – we're not saying that these events really happened in the real world, and this information should not be added to Scone's real-world model. Second, the pigs and wolves in this world-model are anthropomorphic in many respects: they think like humans, the pig built a house from bricks, and so on. But these animal characters also inherit some properties from their real-world models: wolves want to eat pigs; pigs want to stay alive; if the wolf has physical access to the pig, the pig is going to lose the struggle; and so on. The story doesn't provide a lot of details about what other wolf or pig traits are inherited or over-ridden – do they walk on four legs, or upright as in the Disney cartoons? – so we just leave those things unspecified. They are not important to this story.

As for C*PIG1*, C*WOLF1*, and C*WP1*, they are potentially important, but are uninteresting here, since each is a simple clone of C*STORY1*, adding or subtracting nothing that is of interest in this story. In all four contexts, the wolf is waiting outside the house.

If we want to build a system that understands this story, the system must be able to *activate* each of these contexts in turn and reason about the consequences of the knowledge it finds there. In this case, in C*PIG1* and C*WP1*, the pig is able to deduce that it's too dangerous to unlock the door right now – it would likely lead to his death – and that fear is stronger than his desire to go outside. (If the pig were on the verge of starvation, the decision might go the other way.) So, as things stand, the pig is going to keep the door locked as long as the wolf is outside (or at least for a long time), and the wolf knows that as well.

Now we come to the trick or act of deception. As described earlier, the generic {trick} concept is a frame structure in the Scone KB with a number of slots, each restricted to a certain kind of filler. There is, at least, an {agent} slot, which must be filled by some animate being;

the {victim} slot, an animate being with some cognitive capability; a specific {trick-action}, which can be any action taken by the agent, simple or complex (this itself may be a frame with many sub-actions); and a {time of action}, which must be a time-point or interval.

When we read the story, we instantiate this frame and inherit (by default) its slots, restrictions, and other contents. So the wolf is the {agent}, the pig is the {victim}, the business with the foot-stomping is the {trick-action} and the exact time is unspecified – it is just a time-interval contained within the larger time-interval spanned by this part of the story.

Like all subtypes of {event}, the {trick} description also contains slots for the {before-context} and the {after-context}. In the before-context there is a context representing the victim's mental state before the trick-action occurs. This corresponds to reality, at least in some important respect. The goal of the trick-action is somehow to change this mental state to a different one that no longer corresponds to reality, but that is somehow more beneficial to the actor. There are two possible outcomes of this action: either the trick succeeds, and the victim's mental state becomes equivalent to the one that the actor desired, or the trick fails, and the victim's mental state remains (in this respect) unchanged.

So in this instance, the trick-action divides the reality of the story into two parts: the C_{STORY1} context that we saw before, and the C_{STORY2} context – story-reality after the trick occurs. In addition, we have a new mental state for the pig, C_{PIG2}, which lives in C_{STORY2}. In C_{STORY1}, we have a new state, the wolf's representation of the *desired* C_{PIG2}, which we will call C_{GOAL}. In C_{GOAL}, the wolf is gone, and the pig will deduce that it is safe and desirable to unlock the door and go outside.

Since the trick succeeded, C_{PIG2} is indeed the same as C_{GOAl}. We readers can activate this C_{PIG2} and deduce that (in the absence of any surprise plot-twists) the wolf will be dining on pork tonight. The reasoning system could also represent and reason about an alternative ending, in which the pig is less gullible and still believes that the wolf is outside. That is represented by $C_{PIG2\text{-}ALT}$ in the diagram. We can deduce that in this state the pig will remain inside (at least until he becomes very hungry), and will remain safe.

Just to be clear: In the current Scone system, we do not have a general system for the understanding of human language. We have built some small prototype English-to-Scone systems based on construction grammars, but these are not yet capable of understanding the sort of language that we see in this story and others like it. Nor do we have a general problem-solver capable of making all the kinds of deductions outlined above, though we can handle simple rule-chains such as "If a dangerous animal is outside your house, remain inside and keep the door locked".

So Scone, by itself, is not able to understand this story in any general sense. My point here is simply that the manipulation and maintenance of all these mental states is a big part of this problem, or any problem involving trickery and deception, and that Scone's multiple-context mechanism provides an efficient and relatively intuitive platform for representing and reasoning about these states. This representational capability is not a full solution to problems like this, but it is an important enabling condition.

Of course, deception/trickery is not the only type of action whose goal is to alter someone's mental state, or your own. We need much the same machinery to represent and reason about distracting someone, forgetting and reminding, giving directions, taking action to learn something new, and so on. Some of these additional areas are explored in (Chen and Fahlman 2008).

Conclusions

Based on the above account, I believe we can make the following claims:

- The multiple-context mechanism in Scone provides an efficient, flexible, and easy-to-understand representation – not a full solution, but an *enabling technology* -- for a number of tasks that play an important role in human-like thought. These include reasoning about multiple hypotheses and counter-factual situations, maintaining a distinct model of each, but with much shared knowledge; planning and episodic memory, including questions related to the frame problem; and modeling the mental states of other actors in a scenario, including both their beliefs and their knowledge. There is much work left to do in each of these areas, but using a system like Scone avoids some of the difficulties we encounter when using less expressive representations based on First-Order Logic or Description Logic.
- The multiple-context mechanism described here is not an isolated bit of machinery that can be grafted onto any reasoning system. It is a convenient re-packaging of higher-order logic. It depends on Scone's ability to make statements about other statements, and to use this meta-knowledge to control which statements will play a role in the system's subsequent processing. It also makes heavy use of Scone's ability to *cancel* some information that would otherwise be inherited – a form of non-monotonic logic. These capabilities are incompatible with guarantees of logical completeness or provable consistency of the KB.
- In all of this discussion, we are talking about the emulation of human-like abilities. We are making no claims about implementation. Scone makes use of the

serial processing and great speed of modern workstations. A neural implementation would take advantage of massive parallelism and redundancy to compensate for much slower and less reliable components.

- Our description of the human-like processes addressed here is based on simple observation and a bit of introspection, not on specific findings from neuroscience or cognitive psychology. For these large and obvious phenomena, we believe that is sufficient. However, it might be very interesting to investigate whether careful experiments can observe anything like the context-based observations we describe here in human subjects.
- So, I would claim, there is a trade-off here: If we want to perform the kinds of reasoning described in this paper – and certainly if we want to perform that reasoning in a way that can scale up to knowledge base of human-like size – we must be willing to forego logical completeness and provable consistency, and focus instead on more limited and local forms of reasoning. For a system like Scone, whose goal is to support natural language understanding and human-like, "common sense", "good enough" reasoning, we think that's a reasonable trade-off, and probably a necessary one.

References

Chen, W. and Fahlman, S. E. 2008. Modeling Mental Contexts and Their Interactions. AAAI 2008 Fall Symposium on Biologically Inspired Cognitive Architectures.

Fahlman, S. E. 1979. NETL: A System for Representing and Using Real-World Knowledge. MIT Press. Available online at ftp://publications.ai.mit.edu/ai-publications/pdf/AITR-450.pdf.

Fahlman, S. E. 2006. "Marker-Passing Inference in the Scone Knowledge-Base System", in proceedings of the First International Conference on Knowledge Science, Engineering and Management (KSEM'06). Springer-Verlag (Lecture Notes in AI).

Fahlman, S.E. 2008. In Defense of Incomplete Inference. In the Knowledge Nuggets blog: http://cs.cmu.edu/~nuggets/?p=34

Fikes, R. E. and Nilsson, N. J. 1971. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189-208.

Forbus, K. D. 1988. Qualitative Physics: Past, present, and Future. In *Exploring Artificial Intelligence*, Shrobe, H. ed., Morgan Kaufmann.

Hanks, S. and McDermott, D. V. 1987. Nonmonotonic logic and temporal projection. Artificial Intelligence 33(3):379-412.

Lenat, D.B.; Guha, R.; Pittman, K.; Pratt, D.; and Shepherd, M. 1990. Cyc: toward programs with common sense. CACM v30, n8, pp. 30-51.

McCarthy, J. and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Machine Intelligence 4: 463–502. Edinburgh University Press.

Sussman, G. J. and McDermott, D.V. 1972. From PLANNER to CONNIVER: a genetic approach, Proceedings of the 1972 AFIPS Fall Joint Computer Conference, Part 2.