

Towards a Model of Heterogeneous Commonsense Reasoning¹

M. Martinez

¹Paper published in the Proceedings of WoLLIC'2001, J. Baldwin, R. de Queiroz & E. H. Hauesler (eds.), *Matemtica Contemporanea*, V 24, Sociedade Brasileira de Matematica, July 2003.

Abstract

A preliminary version is presented of a model for commonsense reasoning, developed in joint work with Jon Barwise. Among others, the model addresses the question of how the symbolic representations used by a reasoning agent (like words in natural language) are linked to the reality outside the reasoner's "mind". In our framework, a collaborative approach is used, in which the symbolic-linguistic level of representation interacts with a pre-symbolic level given by a state space. The structure of the state-space allows to encode regularities of the world in an implicit (more analogous) manner, just as it is common practice in mathematical modeling in science and more specifically in many models of animal cognition. The interaction between both levels also opens the window to the possibility of simultaneous use of symbolic and numerical tools for inference. Formally, we introduce a logical system based on reduction rules and we show how it relates to classical logical systems based on Gentzen rules. At a more practical level, we present S^3 , an inference engine implemented in *Mathematica*, which has been used as a prototype to test ideas from the formal model and to get new insights to enrich it. Non-monotonicity and context dependence seem to be some of the phenomena that can be addressed in a novel way by using our proposed framework. ¹

¹I am grateful to Larry Moss and to two anonymous referees for their useful comments and suggestions on this paper.

0.1 Introduction

Some high level cognitive abilities, like logic or language, are frequently studied by following a classical symbolic approach. Proof systems, rewriting systems, production rules, finite automata have all proven useful in modeling such kinds of abilities, especially when working with domains for which embodiment is not an issue. That is the case of mathematical domains, where we refer only to abstract objects which inhabit the realm of the mind, not the external physical world that we perceive through our senses. Purely symbolic approaches, however, do have difficulties when trying to capture in a fully satisfactory way phenomena like non-monotonicity in commonsense embodied reasoning. They also pose a difficult question regarding the nature and origin of the connection that must exist between mental representations and the external world. Classical semantical approaches usually relate symbols to their referents in the world, but it is not clear how the connection is created in the first place.

These problems arise when we acknowledge that in modeling a situated agent there is an environment to be taken into account, and that situated reasoning is relative to the agent's knowledge of the world, much of which is acquired from experience. In fact, when it comes to modeling biologically plausible ways in which an agent can get "attuned" to its world by experience, one has to say that more numerical-oriented approaches, like neural networks or dynamical systems, are quite appealing. One reason is that in these models mental representations are points in a (normally multidimensional) state space endowed with certain structure, like a topology or a metric. Much of the structure of the world can then be mapped into the structure of the state space without having to make everything explicit. This is the case with similarities: in a model using a metric state space, similar objects or situations in the world are mapped to close points in the state space, while very dissimilar objects are mapped to distant points. This way, knowledge about similarity relations in the world is available to the agent for its use, even if it is not stated explicitly by any sort of symbolic representation. On the other hand, these models alone seem to be insufficient when trying to obtain from them the powerful insights and tools that classical approaches do provide in areas such as logical reasoning.

The framework presented in this paper (joint work with Jon Barwise) intends to be an initial step towards an account of high-level cognitive abilities like logic and language that integrates the insights provided by both classical logic semantics and the methods of modeling with state spaces used in science. Here we focus on logical reasoning and, more concretely, on the question of how an agent can judge a claim about the world as being valid or invalid, based on its empirically acquired understanding of the world. The rationale behind our proposed solution to this problem includes the following ideas.

- A sophisticated agent needs to manipulate information encoded in different systems of representation. Instead of assuming the existence of an interlingua, we take it that there is no translation of representations within the agent's mind, but rather negotiation processes between differ-

ent systems of representation. This is what we call a *heterogeneous system*. Different systems of representation are justified because of their adequacy in dealing with specific or local tasks. In particular, since symbolic systems have proven useful in modeling (at the very least the non-situated version of) logical reasoning, and since numerical models based on structured state spaces have been successful in modeling the way in which an agent develops mental representations of the world based on experience, we work with a heterogeneous representation system that includes both a symbolic and a pre-symbolic level of representation, the latter based on a structured state space.

- We adhere to the reasons given by Gardenfors in [?] in favor of the existence of a conceptual level that stands somewhere in the middle between the symbolic level (where the structure of a symbol does not resemble the structure of the referent) and the perceptual level (where we have highly-dimensional state spaces that capture lots of fine structure in the world, but whose dimensions have frequently no psychological significance). Our pre-symbolic level corresponds closely to Gardenfors’s conceptual level: our state space has few dimensions, all of them intended to be psychologically significant. Moreover, we use some special families of sets of states whose geometry permits to combine them in useful ways.²
- We borrow ideas from Barwise’s and Seligman’s suggestion that phenomena like non-monotonicity or the frame problem in commonsense reasoning can arise from shifts in the contexts under which logical reasoning takes place (See [?]). Within a fixed context, though, the classical rules of logic, including Weakening, apply as expected. In our framework, the question “Is $\Gamma \vdash \Delta$ valid?” is always formulated in a context that imposes biases in the reasoning process, so that answers to the same question, when asked in different contexts, can be different.
- Our solution reflects the Inverse Relationship Principle which guides the pragmatological theory of modality proposed by Jon Barwise in [?].

Inverse Relationship Principle: Whenever there is an increase in available information there is a corresponding decrease in possibilities, and viceversa.

Following this, the question “Is $\Gamma \vdash \Delta$ valid?” is approached by gradually shrinking the set of situations that could possibly falsify the sequent.

Our work so far has yielded two kinds of results. First, an inference engine implemented in *Mathematica*, which we have called S^3 (for Structured State Spaces). We present it in Sections 2 and 3, stressing knowledge representation and the algorithm for testing whether a claim about the world (a sequent) is

²However, the class of subsets of a state space from which Gardenfors’ takes concepts is different to ours. The operations by which special sets are combined are also different.

valid in a certain context. The second kind of result, the topic in Section 3, is a proof system based on reduction rules which act on (sets of) extended sequents. The “extended sequents” used in both S^3 and the formal proof system reflect the hybrid nature of knowledge representation, since they include a special subset of the state space which represents a region of the state space, a set of possible situations, where attention is focused on. The main result here compares our reduction system with a more standard proof system based on Gentzen rules, and says that as far as standard deduction goes (using Cut, Weakening and Identity), our reduction system performs as well as possible. One reason to prefer the reduction system to the standard Gentzen rules is that the former better reflects the Inverse Relationship Principle.

0.2 The S^3 inference engine.

The S^3 inference engine, implemented in *Mathematica*, is a system that allows us to model physical systems and assess the validity of claims about such systems. Although in principle any sort of simple physical system could be modeled (thermostats, elevators...) we are especially interested in the case where the model is intended to be the model an agent might maintain about its world in order to reason about it. We will continue using intuitions associated to the modeling of an agent’s knowledge, although the reader should keep in mind that our methodology is more general. This section is intended to give the reader a basic understanding of S^3 . The next section contains several concrete examples that illustrate what is described in this one.

In S^3 knowledge representation involves two levels. At what we call the symbolic level, there is a simple vocabulary of *types* with which assertions about the system can be made. The second level, the pre-symbolic level, is based on a product space with some additional structure, where the dimensions are features of the world which are perceived by an agent (or measured by a scientist in the more general case). A basic intuition linking the two levels is that in order to reason about the environment an agent has learnt some *concepts* (special subsets of the state space) by which it categorizes situations, and each important concept has a *type* that names it. We will use letters A, B, C, \dots for types and uppercase Greek letters $\Gamma, \Delta \dots$ for sets of types.

To have a more concrete setting to think about, imagine a simple robot, Rob, which is able to reason about a bulb controller which consists of only two parts: a bulb and a slider by which one can change the intensity of light. When the slider is completely up, the bulb intensity is maximum, and when the slider is completely down, the bulb is off, and in general the intensity increases as the slider is pulled up. There are four things Rob can observe or perceive. He can perceive the current slider’s position, how much natural light enters the room and the overall level of illumination in the room at a given moment. Rob can also observe something which is not an aspect of the external world, but rather an internal preference. Namely, at each moment Rob has a favorite level of illumination that he would like to perceive (as the total of the natural light and

artificial light contributions). Rob is able to change the position of the slider, and he does so guided by his current preferences³.

In order to work, S^3 is fed with a model, which is provided by the user and consists of a symbolic and a pre-symbolic part. The pre-symbolic part of a model in S^3 is the definition of a state space $\Omega = \prod_{i=1}^n \text{dom}(d_i)$, where the observables d_1, \dots, d_n are basic features of the world which an agent (or scientist) \mathfrak{A} is able to perceive and $\text{dom}(d_i)$ is the set of all possible values for d_i . In S^3 , $\text{dom}(d_i)$ is always a finite union of intervals in the real line. The set *Observables* of observables is partitioned into two subsets: the set J of *input observables* and the set O of *output observables*. The value of an input observable is independent of the values of the other input observables, while the value of an output observable is a function of the input observables. The *normal state space* Ω_0 is the set of all those n -tuples in Ω that satisfy the regularities described by the output functions. This kind of knowledge would allow an agent to discriminate between states in Ω which are realizable (those in Ω_0) and those which are not realizable but still conceivable (those in $\Omega \setminus \Omega_0$).

So in the case of Rob, we use pre-symbolic (vectorial) representations of the form

$$\langle \textit{slider}, \textit{natural light}, \textit{preference}, \textit{room illumination} \rangle$$

to characterize particular configurations of the world. Rob’s “understanding” of the world is based then in four basic features. If we suppose that the status of each feature in a particular moment can take values between 0 and 1, then an example of the state corresponding to a particular configuration of the world could be a tuple like $\langle 0, 0.8, 0.8, 0.8 \rangle$, meaning that the slider is completely down, the level of natural light is 0.8 (quite good), Rob’s current preference is 0.8 (so he happens to be happy with the current situation) and the level of light in the room is 0.8 (since only the natural light contributes). A S^3 model for Rob’s pre-symbolic understanding about its world is summarized in the following table, together with the function F .

Observable	Name	Kind	Domain
Slider position	<i>slid</i>	input	$[0, 1]$
Natural light	<i>nat</i>	input	$[0, 1]$
Preference	<i>goal</i>	input	$[0, 1]$
Room illumination	<i>inten</i>	output	$[0, 1]$

with $\textit{inten} = F(\textit{slider}, \textit{nat})$, where $F(x, y) = \max(x^2, y)$.

The state space here is $\Omega = [0, 1]^4$ and the normal state space is

$$\Omega_0 = \{ \langle x, y, w, z \rangle \in \Omega \mid z = F(x, y) \}.$$

At the pre-symbolic level, given the definition of particular Ω and Ω_0 , S^3 deals with representations for single *partial* states in Ω and for simple subsets

³...although we will not speak about actions here. We don’t worry either about how Rob’s preferences change with time.

of Ω , called *basic sets*. The class \mathfrak{B} of basic sets, is the smallest class of subsets of $\bigcup_{n=1}^{\infty} \mathfrak{R}^n$ which contains the intervals in \mathfrak{R} and is closed under finite unions, complements, cartesian products and projections. Equivalently, \mathfrak{B} is the class containing all finite unions of “boxes”. The reason why we need partial states and basic sets is that they make it possible to model things such as reasoning about approximate information. For example, we might imagine Rob being able to notice that at certain moment the slider is positioned somewhere between 0.3 and 0.7, without knowing the exact position, and still being able to make useful inferences from this piece of approximate information.

What about the symbolic level of representation? At this level we give to S^3 a simple language \mathfrak{L} with basic lexicon L . The elements of L are called *basic types*. A basic type is intended to be a way to classify situations⁴. The link between a type (as understood by an agent) and a real-world situation is provided by the state space. A situation is represented by a partial state. Therefore, the extension of a type A , which is a set of situations, corresponds to the subset of Ω formed by states that represent situations of type A . We call these special subsets of Ω *concepts*. In S^3 to each type A we associate its concept by defining a function $\chi_A : \Omega \rightarrow \{0,1\}$, such that for all $\sigma \in \Omega$, $\chi_A(\sigma) = 1$ if and only if σ is of type A . The function χ_A may actually depend only on a subset of observables. We call that subset $LivesOn(A)$ and we say that A *lives on* $LivesOn(A)$. The intuition is that when using A , an agent’s attention is focused on the set of dimensions $LivesOn(A)$.

As an example, Rob’s basic lexicon could be

$$L = \{\text{bulbInten}[a], \text{wantLight}[\text{more}], \text{wantLight}[\text{equal}], \text{wantLight}[\text{less}], \text{useful}\}$$

where *bulbInten*[a] stands for “the bulb brightness is a ” (a a real number between 0 and 1), *wantLight*[p] stands for “I want more/less/equal light” (depending on what p is) and “useful” stands for “I think that the bulb’s light is useful”. We will set things up to model the case where Rob thinks that the bulb’s light is useful when the two following conditions hold: (1) the bulb’s intensity is at least .1 more than the intensity of natural light, and (2) the bulb’s intensity lies in the range Rob likes. The table below presents some examples of reasonable concepts associated with Rob’s types:

Type A	Concept Ω_A
bulbInten[a]	$\{\sigma \in \Omega \mid \text{slid} \in (a - 0.05, a + 0.05)\}$
wantLight[less]	$\{\sigma \in \Omega \mid \text{inten} + 0.1 \leq \text{goal}\}$
wantLight[equal]	$\{\sigma \in \Omega \mid \text{inten} - \text{goal} < 0.1\}$
wantLight[more]	$\{\sigma \in \Omega \mid \text{inten} - 0.1 \geq \text{goal}\}$
useful	$\{\sigma \in \Omega \mid \text{slid}^2 > \text{nat} + 0.1 \wedge \text{goal} - \text{inten} < 0.1\}$

Observe that, Ω_A can very well be, as in the case of $\Omega_{\text{bulbInten}[0.5]}$, a “simple” basic set or it could be, as in the case of Ω_{useful} , a non-trivial subset of Ω . In

⁴A situation is just a part of the world. Situation semantics is a form of model theory in which, instead of possible worlds, possible situations are used. Situation theory arose, among other things, from trying to acknowledge the importance of context in semantics. See [?] for the original formalization of situation semantics

general then, it is important to note that there might be Ω_A 's which are not basic sets.

Although we have argued above that basic sets are useful even when working only at the pre-symbolic level, the main use for them in S^3 is to provide the link between symbolic and pre-symbolic representations that allow both levels to interact in useful ways. So in S^3 to each type A we also assign functions, $Nec_A, Suf_A : \mathfrak{B} \rightarrow \mathfrak{B}$, such that for all $\Phi \in \mathfrak{B}$ and all $\sigma \in \Omega$,

$$\begin{aligned} Nec_A(\Phi) &\subseteq \Phi, \\ Suf_A(\Phi) &\subseteq \Phi, \\ \sigma \in Suf_A(\Phi) &\longrightarrow \sigma \in \Omega_A \cap \Phi, \text{ and} \\ \sigma \notin Nec_A(\Phi) &\longrightarrow \sigma \notin \Omega_A \cap \Phi. \end{aligned}$$

The idea is that for each possible ‘‘window’’ given by a basic Φ , $Nec_A(\Phi)$ and $Suf_A(\Phi)$ approximate the ‘‘shape’’ of the region Ω_A in terms of basic sets.

S^3 is an inference engine. Once a model has been provided by the user, consisting of both a symbolic and a pre-symbolic part, S^3 uses this knowledge in order to solve *queries* of the form

$$(1) \quad \text{Is } \Gamma \vdash \Delta \text{ valid in the context } \Psi?$$

A query consists of a *sequent* $\Gamma \vdash \Delta$ and a *context* Ψ , which is a product of 1-dimensional basic sets encoding information available to the agent regarding the possible values for each observable. For example, in the case of Rob, one might have a context $DarkOutside = [0, 1] \times [0, 0.3] \times [0, 1] \times [0, 1]$. This does not encode any special information about most dimensions, except that natural light is very dim.

The process of trying to solve a query in S^3 has two stages. First, the query above is transformed into an *extended sequent* $\Gamma \vdash \Delta$ (*on* Φ), where Φ is a basic set that contains the context Ψ and depends on both Ψ and $\Gamma \vdash \Delta$. The second stage tries to find out whether the extended sequent $\Gamma \vdash \Delta$ (*on* Φ) is valid. The extended sequent is valid only if every state $\sigma \in \Omega_0$ that satisfies *all* the types in Γ , satisfies *at least* one of the types in Δ . It is in testing for the validity of extended sequents that the functions Nec and Suf are used once and again, as we will see. We should stress that Ω_0 , the normal state space, might not be a basic set. Here, however, we will assume that Ω_0 is basic, and even more, that $\Omega_0 = \Omega$, in order to facilitate the discussion⁵.

Stage 1: From queries to extended sequents.

In a query like (1), we can think of the context Ψ as an encoding of *default assumptions* about the values of the different observables in the moment the query is made. On the other hand, when we want to test for the validity of $\Gamma \vdash \Delta$, we can look at Γ as forcing us to consider as set of possible counterexamples a set of states that is more general than Ψ , for in order to assess the

⁵The issue of the non-constructibility of Ω_0 in some cases is interesting in the sense that it is another point of the framework where continuous methods seem to be useful. We have not really gone deeply into this issue.

validity of $\Gamma \vdash \Delta$, one needs to consider all the possible (partial) states that satisfy Γ . How is this done in S^3 ? Well, in order to test if a state σ is of type A , one looks only at the values of dimensions in $LivesOn(A)$. Therefore, it is natural to say that if one wants to take into account all of the possible configurations regarding Γ , one should free any constraint that the current context $\Psi = \prod_{i=1}^N I_i$ may be imposing on dimensions in $\bigcup_{A \in \Gamma} LivesOn(A)$, and then test whether the sequent holds of all states in that enlarged basic set Φ . Formally, S^3 transforms the query **(1)** into the extended sequent $\Gamma \vdash \Delta$ (*on* Φ), with $\Phi = \prod_{i=1}^N J_i$, where $J_i = dom(d_i)$ if $d_i \in \bigcup_{A \in \Gamma} LivesOn(A)$ or if d_i is an output dimension, and $J_i = I_i$ otherwise.

It is in this stage where phenomena like non-monotonicity can arise, because a question about $\Gamma \vdash \Delta$ made in two different contexts Ψ_1, Ψ_2 , might lead to two different extended sequents with conditions Φ_1, Φ_2 . It might very well happen that only one of the extended sequents is valid. In the examples we have implemented in S^3 such a situation occurs when some of the *LivesOn* dimensions for Γ are *output* dimensions. Those output dimensions might depend on some other input dimensions for which the constraints imposed by the original contexts Ψ_i are unchanged when we move from the context Ψ to the extended conditions Φ_i , $i=1, 2$. We look at this as a way to model the role of attention in commonsense reasoning. That is, for us the *LivesOn* dimensions are attentional parameters that impose biases in the reasoning process, and which depend on the particular query to be solved, as will be illustrated later, with concrete examples, in the next section.

Stage 2: Testing the validity of an extended sequent.

The product of the first stage is an extended sequent $\Gamma \vdash \Delta$ (*on* Φ) for which S^3 should try to find out whether it is valid or not. An extended sequent is valid only if there is no counterexample for it. S^3 tries to show that an extended sequent is valid by showing that the set of possible counterexamples is empty. To do that, it identifies a set that must contain all possible counterexamples for the sequent, and it starts a process of gradual shrinking that at each step throws away states which are discovered to satisfy the sequent. The hope is that after finitely many shrinking steps the empty set will be reached. If that is the case, the sequent has been shown to be valid, for there is no counterexample that might be found for it. If the empty set is not reached, then a narrowing of the initial set has been performed that at least would allow the system to make a more informed search for counterexamples in a third stage of the process⁶. Here is how the shrinking process in S^3 works.

It follows from the definition of validity of an extended sequent that $\Gamma \vdash \Delta$ (*on* Φ) is *not* valid if and only if t

$$(2) \quad \left(\bigcap_{A \in \Gamma} (\Omega_A \cap \Phi) - \bigcup_{B \in \Delta} (\Omega_B \cap \Phi) \right)$$

is *not* empty, where Ω_A is the set of states of type A ⁷ Observe that when all

⁶This third stage is not fully described here, for it is quite undeveloped so far.

⁷In fact, we should have said that the sequent is not valid if and only if the intersection of

the Ω_B 's and Ω_A 's are basic sets, we can check for validity directly by using (2), since \mathfrak{B} is closed under intersection and complementation. The situation is more complicated when some of the Ω_A 's (or Ω_B 's) are non-basic. However, in these cases we still can use the functions Nec_A and Suf_A by defining the operator

$$(3) \quad Nec(\Gamma \vdash \Delta \text{ on } [\Phi])_{Suf} =_{def} \bigcap_{A \in \Gamma} Nec_A(\Phi) - \bigcup_{B \in \Delta} Suf_B(\Phi).$$

because it is an easy observation that any counterexample to $\Gamma \vdash \Delta$ (on Φ), if there is any, must also belong to the set

$$\Phi_1 =_{Nec} (\Gamma \vdash \Delta \text{ on } [\Phi])_{Suf},$$

although not necessarily every element of Φ_1 is a counterexample to $\Gamma \vdash \Delta$ (on Φ). In other words, it is always true that $\Gamma \vdash \Delta$ (on Φ) is valid if and only if the sequent $\Gamma \vdash \Delta$ on Φ_1 is valid, where $\Phi_1 \subseteq \Phi$ is the set defined above. But now we can iterate the reduction process in order to obtain a third sequent $\Gamma \vdash \Delta$ on Φ_2 which is valid if and only if $\Gamma \vdash \Delta$ on Φ_1 is valid, and so on⁸. If we let $\Phi_0 = \Phi$, at step $n + 1$ we obtain a sequent $\Gamma \vdash \Delta$ (on Φ) on Φ_{n+1} , with

$$\Phi_{n+1} =_{Nec} (\Gamma \vdash \Delta \text{ on } [\Phi_n])_{Suf}.$$

and such that the set of counterexamples for this sequent is exactly the set of counterexamples for the original sequent $\Gamma \vdash \Delta$ (on Φ). An upper bound M is imposed for the number of iterations of this shrinking process that are allowed in S^3 , since there is no guarantee that a fixed point exists for an arbitrary sequent (unless certain conditions on the Nec and Suf functions are met). That is, the process stops as soon as S^3 reaches an N such that $\Phi_N = \Phi_{N-1}$, or $N = M$, whichever occurs first.

There are four cases to consider after the iteration process in S^3 has finished.

1. $\Phi_N = \emptyset$. In this case S^3 says that the sequent is valid.
2. Φ_N is a finite set. S^3 uses the functions χ_A , which define the meaning of the types, to test whether each one of the states in Φ_N satisfies the sequent. The system says that the sequent is valid if no counterexample is found. Otherwise, it exhibits the first counterexample it finds.
3. Φ_N is infinite and the set $\Theta = \bigcap_{A \in \Gamma} Suf_A(\Phi_N) - \bigcup_{B \in \Delta} Nec_B(\Phi_N)$ is non-empty. Notice that Θ is **not** the same as the set Φ_{N+1} because Nec and Suf are shifted. Since every state $\sigma \in \Theta$ is a counterexample for $\Gamma \vdash \Delta$ (on Φ_N), the system answers that the original sequent is not valid.

(2) and Ω_0 is *not* empty. However, since we are taking Ω_0 to be basic here, this additional checking is not problematic, and we will ignore it through this section.

⁸We have presented here a simplified version of the reduction process, where only the contexts change, not the sequents. This is not the case in S^3 though, where sequents do change due to another use of necessary and sufficient conditions not explained here.

4. Φ_N is infinite and the set Θ is empty. Here S^3 cannot tell, based only on the *Nec* and *Suf* functions, whether the original sequent is valid or not. S^3 triggers a search for counterexamples in Φ_N , but so far this search is quite blind. We believe that some additional structure of the state space (like probability distributions for the dimensions) should be used to guide the search and ideally give some measure of the strength of the belief on sequent for which no counterexamples are found.

0.3 Some concrete examples using S^3

The goal in this section is to show how *Mathematica* notation is used in S^3 and to illustrate with a few examples (still concerning the robot example) how S^3 behaves. This section is designed to appear just like a *Mathematica* notebook that uses S^3 . In what follows, text that is enclosed by frames indicates *Mathematica* notation. Each frame corresponds to what is called a *cell* in *Mathematica*. Cells with boldface text represent *input cells*, while cells with regular text indicate *output cells*.

0.3.1 Defining the grounding state space Ω

The first thing we need to do in order to implement an example in S^3 is to load a few packages (that is, to load S^3 itself). To do this, it is necessary to evaluate the following cell.

```
Off [ General :: spell1 ];
Get [ "SSSIE 'BasicSets' " ];
Get [ "SSSIE 'Pdd' " ];
Get [ "SSSIE 'StateSpaces' " ];
```

Remember that Rob's "understanding" of the world is based on four basic features and that each state will have the generic form

$$\sigma = \langle \textit{slider}, \textit{natural light}, \textit{preference}, \textit{room illumination} \rangle$$

A S^3 model for Rob's pre-symbolic understanding about its world that captures our previous description is given to S^3 as follows.

First, we need to define the state space Ω . In so doing, we begin by defining domains and labeling them, one for each dimension of the state space. This is done by defining a matrix with three columns and as many rows as there are dimensions in the space, in this example, four. We also tell how many of these dimensions are input dimensions, three in our case. By convention, these input dimensions always come first in the matrix. The first column contains the labels we will use in displaying values of states, the second contains the name of the variable used by *Mathematica* to refer to the various dimensions, and the third column defines the domain of valid values for each one of those variables.

```

dimensionsAndDomains =  $\left( \begin{array}{lll} \text{Slider position} & \textit{slid} & [0, 1] \\ \text{Natural light} & \textit{nat} & [0, 1] \\ \text{Preference} & \textit{goal} & [0, 1] \\ \text{Room illumination} & \textit{inten} & [0, 1] \end{array} \right);$ 
  

number of Inputs = 3;

```

We feed the information on dimensions given above to the general system by means of the command *SetObservables*, used as follows.

```

SetObservables[labelsAndDomains, numberOfInputs];

```

Here is the definition of the output function that allows to calculate the value of our output dimension *inten*, given the values of the input dimensions.

```

intensityFunction := Max[slid2, nat];

```

We say now that the function above is the one that corresponds to the up now the output dimension *inten*. When more output dimensions exist, a matrix with as many rows as output dimensions must be created.

```

outputFunctions = ( inten intensityFunction )

```

This matrix is passed as the parameter to the command *SetOutputFunctions*, which makes S^3 create internally the associations between output dimensions and their corresponding functions.

```

SetOutputFunctions[outputFunctions];

```

0.3.2 Defining the propositional language and grounding it on Ω

Let's remember Rob's basic lexicon:

- **bulbInten[a]**: The bulb brightness is *a* (with a tolerance of plus or minus 0.05).

- **wantLight[less]:** The robot’s preference is lower than the overall intensity of light (with a tolerance of 0.1).
- **wantLight[equal]:** The robot’s preference coincides with the overall intensity of light (with a tolerance of 0.1).
- **wantLight[more]:** The robot’s preference is higher than the overall intensity of light (with a tolerance of 0.1).
- **useful:** We will set things up to model the case where Rob “thinks” that the bulb’s light is useful when the two following conditions hold: (1) the bulb’s intensity is at least 0.1 more than the intensity of natural light, and (2) the bulb’s intensity lies in the range of brightness that Rob likes.

In what follows, we show the formal definition for these types is given to S^3 . For each type, its definition includes the characteristic function, the necessary and sufficient conditions. In S^3 it is necessary also to declare the type as a valid one that can be used in forming queries. So, for a type α , a function $validTypeQ[\alpha]$ with boolean values is defined in such a way that the value *True* is returned only if its argument is a syntactically correct atomic type. For most types, the function is very simple, basically always returning *True*. However, in cases like the family of types $bulbInten[a]$, an extra check must be made so to ensure that a is an admissible value (in our case a value in $[0, 1]$). Here are then the definitions of our types in S^3 .

$$\begin{aligned}
\mathbf{validTypeQ[bulbInten[a]]} &:= \mathbf{0 \leq a \leq 1}; \\
\mathbf{meaning[bulbInten[a]]} &:= \mathbf{a - 0.05 < inten < a + 0.05}; \\
\mathbf{necessary_{bulbInten[a]}[\Phi_-]} &:= \mathbf{Shrink[\Phi,} \\
&\quad \mathbf{inten \in OpenBasicInt[a - 0.05, a + 0.05]]}; \\
\mathbf{sufficient_{bulbInten[a]}[\Phi_-]} &:= \mathbf{Shrink[\Phi,} \\
&\quad \mathbf{inten \in OpenBasicInt[a - 0.05, a + 0.05]]};
\end{aligned}$$

Notice that in the definition of $bulbInten[a]$ given above, the “*meaning*” function for the type, is simply the characteristic function for it. As for the necessary and sufficient conditions, which in this case coincide, the definitions above say that when Φ is a boolean set,

$$Nec_{bulb[inten[a]]}(\Phi) = \{\sigma \in \Phi \mid inten \in (a - 0.05, a + 0.05)\}.$$

The command $Shrink[\Phi, wff]$ simply calculates the intersection of the boolean set Φ and the boolean set defined by the well formed formula wff . The commands $OpenBasicInt[min, max]$ and $ClosedBasicInt[min, max]$ (that we’ll use below) represent the open interval (min, max) and the closed interval $[min, max]$, respectively. Now we can define the “*wantLight*” types, as follows.

```

validTypeQ[wantLight[more]] := True;

meaning[wantLight[more]] := inten + 0.1 ≤ goal;

necessarywantLight[more][Φ-] := Shrink[Φ,

    goal ∈ ClosedBasicInt[BasicInf[Project[Φ, inten]] + 0.1, ∞] ∧

    inten ∈ ClosedBasicInt[-∞, BasicSup[Project[Φ, goal]] - 0.1]];

sufficientwantLight[more][Φ-] := If [

    BasicSup[Project[Φ, inten]] + 0.1 ≤ BasicInf[Project[Φ, goal]],

    Φ, False];

```

In the definition of *wantLight[more]*, it should be clear that the *meaning* function above captures exactly what the intended meaning of the type is, according to our informal description of it. As for the necessary and sufficient conditions, the code above tells us that

- In order to calculate *necessary* conditions for the type *wantLight[more]* to hold inside the boolean set Φ , observe that any state $\sigma \in \Phi$ that satisfies this type must comply with two constraints. First, the value $goal_\sigma$ must be at least 0.1 larger than the minimum possible value that the *inten* dimension can take in Φ (which is the g.l.b. of the projection $\pi_{inten}(\Phi)$). Second, the value $inten_\sigma$ must be at least 0.1 smaller than the maximum possible value that the *goal* dimension can take in Φ (which is the g.l.b. of the projection $\pi_{goal}(\Phi)$).
- As for sufficient conditions, if it happens that the projection $\pi_{inten}(\Phi)$ of Φ on the dimension *inten* is a basic set that is completely at the left (in the real line) of the basic set $\pi_{goal}(\Phi)$ by a distance of at least 0.1, then clearly the whole set Φ consists of states that satisfy *wantLight[more]*. In any other case, the function for sufficient conditions above returns simply the empty set (which is described by the well formed formula *False*).

The definitions of *wantLight[less]* and *wantLight[equal]* have the same flavor.

```

validTypeQ[wantLight[less]] := True;

meaning[wantLight[less]] := inten - 0.1 ≥ goal;

necessarywantLight[less][Φ-] := Shrink[Φ,
    inten ∈ ClosedBasicInt[BasicInf[Project[Φ, goal]] + 0.1, ∞] ∧
    goal ∈ ClosedBasicInt[-∞, BasicSup[Project[Φ, inten]] - 0.1]];

sufficientwantLight[less][Φ-] := If [
    BasicSup[Project[Φ, goal]] + 0.1 ≤ BasicInf[Project[Φ, inten]],
    Φ, False];

```

```

validTypeQ[wantLight[equal]] := True;

meaning[wantLight[equal]] := Abs[inten - goal] < 0.1;

necessarywantLight[equal][Φ-] := Module[{temp, temp1},
    temp = BasicIntersection[Project[Φ, goal], Project[Φ, inten]];
    temp1 = OpenBasicInt[BasicInf[temp] - 0.1, BasicSup[temp] + 0.1];
    Shrink[Φ, goal ∈ temp1 ∧ inten ∈ temp1]];

sufficientwantLight[equal][Φ-] := Module[{temp, temp1},
    pr1 = Project[Φ, goal]; pr2 = Project[Φ, inten];
    If [Abs[BasicInf[pr1] - BasicSup[pr2]] < 0.1 &&
        Abs[BasicInf[pr2] - BasicSup[pr1]] < 0.1,
    Φ, False];

```

In the definition of the type *wantlight[equal]*, a few *Mathematica* notation observations are necessary here. The absolute value function is denoted by *Abs*. Also, the *Mathematica* construct *Module[list, body]* is used, where *list* is a list of local variables and *body* is a sequence of commands separated by semicolons. The value of the last command is the value of the construct. Finally, && is the

Mathematica notation for the boolean *and* operator. The \wedge notation is our own extension to the system, and is only to be used in our own defined well formed formulas that describe boolean sets. Finally, here is the definition of the type *useful*.

```

validTypeQ[useful] := True;

meaning[useful] := slid2 > nat + 0.1 && Abs[goal - inten] < 0.1;

necessaryuseful[ $\Phi$ -] := Shrink[necessary[wantLight[equal]] [ $\Phi$ ],

    slid  $\in$  ClosedBasicInt[ $\sqrt{\mathbf{BasicInf}[\mathbf{Project}[\Phi, \mathbf{nat}]] + \mathbf{0.1}}, \infty$ ]  $\wedge$ 

    nat  $\in$  ClosedBasicInt[ $-\infty, \mathbf{BasicSup}[\mathbf{Project}[\Phi, \mathbf{slid}]^2 - \mathbf{0.1}]$ ];

sufficientuseful[ $\Phi$ -] := Shrink[sufficient[wantLight[equal]] [ $\Phi$ ],

    If[BasicSup[Project[ $\Phi$ , nat]] + 0.1  $\leq$  BasicInf[Project[ $\Phi$ , slid]]2

     $\Phi$ , False];

```

0.3.3 Contexts

S^3 provides a couple of different ways to set context assumptions, but we'll only use the *ChangeContext* command here. As with any other command in S^3 (and any command in *Mathematica*), you can access a brief description of the command *ChangeContext* by writing its name preceded by a question mark.

?ChangeContext

ChangeContext[[{*dim*₁, *basic*₁}, ..., {*dim*_{*k*}, *basic*_{*k*}}] modifies the current context by changing the domains of some of the dimensions as indicated by the argument. For all the other dimensions the domains are unchanged.

Here are the commands that we'll use in this example.

```

Preference[a_] := ChangeContext[{{goal, ClosedBasicInt[a, a]}}];

SetSlider[a_] := ChangeContext[{{slid, ClosedBasicInt[a, a]}}];

DarkOutside := ChangeContext[{{nat, ClosedBasicInt[0, 0.25]}}];

```

0.3.4 Entailment examples.

We will test the following sequents using a few different contexts.

$$\begin{aligned} \text{Sequent}_1 &: \quad \text{wantLight}[more] \vdash \neg \text{wantLight}[less]. \\ \text{Sequent}_2 &: \quad \text{bulbInten}[0.4] \vdash \neg \text{wantLight}[equal]. \\ \text{Sequent}_3 &: \quad \text{bulbInten}[0] \vee \text{bulbInten}[1] \vdash \neg \text{wantLight}[equal]. \end{aligned}$$

Queries using the default context.

The default context is the one where no constraints are imposed on the dimensions. It can be set in S^3 as follows.

ClearContext; ShowContext;

$$\left(\begin{array}{ll} \text{slid} & [0, 1] \\ \text{nat} & [0, 1] \\ \text{goal} & [0, 1] \\ \text{inten} & [0, 1] \end{array} \right)$$

The command used to input the query $\Gamma \vdash \Delta$ in S^3 is $ValidQ[Sequent[\Gamma, \Delta]]$. Here is how S^3 behaves when asked about the validity of the three above sequents.

ValidQ[Sequent[{"wantLight[more]"}, {"wantLight[less]"}]];

The sequent is valid because it reduces to $\{\text{wantLight}[less], \text{wantLight}[more]\} \vdash \{\}$ constrained on the empty set."

The change from the original sequent to $\{\text{wantLight}[less], \text{wantLight}[more]\} \vdash \{\}$ results from eliminating the \neg connective by applying the usual Gentzen rules for natural deduction.

The two last sequents are found to be invalid, and counterexamples are shown.

ValidQ[Sequent[{"bulbInten[0.4]"}, {"wantLight[equal]"}]];

After investigating only 1 critical case for the sequent, I found

	<i>Slider</i>	0
the counterexample	<i>NaturalLight</i>	0.4
	<i>Preference</i>	0
	<i>Intensity</i>	0.4

ValidQ[Sequent[{"bulbInten[0] \vee bulbInten[1]"}, {" \neg wantLight[equal]}]]];

After investigating only 1 critical case for the sequent, I found
the counterexample

<i>Slider</i>	0
<i>NaturalLight</i>	0
<i>Preference</i>	0
<i>Intensity</i>	0

Judging sequents under a different context.

We can modify the current context (which is the default we used in the previous example) as follows.

DarkOutside; Preference[0.2]; ShowContext;

$$\left(\begin{array}{ll} \textit{slid} & [0, 1] \\ \textit{nat} & [0, 0.25] \\ \textit{goal} & \{0.2\} \\ \textit{inten} & [0, 1] \end{array} \right)$$

These are the results when we ask S^3 to judge the validity of the same three sequents.

ValidQ[Sequent[{"wantLight[more]"}, {"wantLight[less]}]]];

The sequent is valid because it reduces to
 $\{wantLight[less], wantLight[more]\} \vdash \{\}$ constrained on the empty set."

No matter in which context we ask S^3 to evaluate the previous sequent, since the sequent is globally true, there is no way to come up with a counterexample for it, regardless of the context.

ValidQ[Sequent[{"bulbInten[0.4]"}, {"wantLight[equal]}]]];

After investigating only 1 critical case for the sequent, I found
the counterexample

<i>Slider</i>	0.6
<i>NaturalLight</i>	0
<i>Preference</i>	0.2
<i>Intensity</i>	0.36

Observe that although both with the default context and our new current context S^3 finds counterexamples for this second sequent, the counterexamples are different. By looking at the characteristic function of $bulbInten[0.4]$ the reader can see that $LivesOn_{bulbInten[0.4]}$ consists only of the dimension $inten$, and therefore this is the only dimension for which constraints are dropped when starting the reasoning process. The constraints on all other dimensions are kept as defaults, and so our counterexample here is one that complies with the two conditions imposed by the current context on the dimensions nat and $goal$: Notice that our counterexample is one where $nat = 0$ (it is therefore dark outside), and $goal = 0.2$.

Finally, by comparing the results for the third sequent when evaluated in the default and our new current context, the reader can see how nonmonotonicity effects appear in S^3 . While it is possible to easily find a counterexample for the sequent when working in the default context (when thinking in a non-situated way), it is not possible to find such a counterexample when the defaults on the dimensions nat and $goal$ are kept. notice, again, that these defaults are not dropped because these dimensions do not belong to $LivesOn_{bulbInten[a]}$ for any a .

ValidQ[Sequent[{"bulbInten[0] \vee bulbInten[1]"}, {" \neg wantLight[equal]"}]]];

The sequent is valid. It reduces to a list of simple sequents which are valid because they are constrained on the empty set. The list of simple sequents is $\{\{bulbInten[0], wantLight[equal]\}, \{\}, \{bulbInten[1], wantLight[equal]\}, \{\}\}$

After dropping any default assumptions on dimensions of $LivesOn_{bulbInten[0]}$ and $LivesOn_{bulbInten[1]}$ (which in this case amounts to doing nothing), S^3 eliminates boolean connectives by using the usual Gentzen rules for natural deduction. Indeed, the sequent

$$bulbInten[0] \vee bulbInten[1] \vdash \neg wantLight[equal]$$

reduces this way to the pair of sequents

$$bulbInten[0], wantLight[equal] \vdash \{\}$$

$$bulbInten[1], wantLight[equal] \vdash \{\}$$

Since both of these sequents are judge as valid by S^3 in our current context, so is the original sequent.

A final example.

Suppose now that we work in a context where it is still dark outside, but the robot's preference for light is 0.9 and the slider's position is 0.9.

Preference[0.9]; SetSlider[0.9]; ShowContext;

$$\begin{pmatrix} \textit{slid} & \{0.9\} \\ \textit{nat} & [0, 0.25) \\ \textit{goal} & \{0.9\} \\ \textit{inten} & [0, 1] \end{pmatrix}$$

This is what we get if we now ask S^3 about the validity of $\textit{sequent}_2$.

ValidQ[Sequent[{"bulbInten[0.4]"}, {"wantLight[equal]}]]];

The sequent holds of all of the 101 critical cases I checked so it is probably valid in the current context. ”

This example shows again nonmonotonicity (compare with the results for this sequent using the other contexts). Observe that the answer S^3 gives here is not a definite one. The reason is that the set of possible counterexamples for this sequent could not be *proven* to be empty, so a sampling of states was made for testing. Since all of those states satisfy the sequent, S^3 judges the sequent as “probably valid”.

0.4 Reasoning with sequents

We are interested in formalizing, generalizing and enriching the ideas which have been already implemented in S^3 . This section presents a first step in this direction. Namely, we study ways to re-use sequents which have been previously proved to be valid (this is something that S^3 does not do currently). Ideally, such sequents should be added to the resources provided by the *Nec* and *Suf* functions when trying to prove a new sequent.

So through this section we postulate the existence of a set *Givens* of extended sequents, which acts as a set of axioms. We study how our reduction strategy can use such a general set of axioms and we ask ourselves how we can compare the framework that we will obtain with more standard proof systems. Two preliminary remarks are in order. First, regarding ways to link standard proof systems and our framework here, it is an easy observation that the standard rules of Identity, Weakening and Cut can be naturally formulated in terms of extended sequents. For example, the Cut Rule looks as follows

$$\text{(Standard Cut)} \quad \frac{\Gamma, A \vdash \Delta \text{ (on } \Phi) \text{ and } \Gamma \vdash \Delta, A \text{ (on } \Phi)}{\Gamma \vdash \Delta \text{ (on } \Phi)}.$$

The second remark has to do with the relationship between this section and the previous one. After reading this section, it should be an easy exercise for the reader to see that the procedure to shrink condition Φ_N used by S^3 can be seen as a particular case of the procedure described in this one, when we set

$$\begin{aligned} \text{Givens} &= \{ \Lambda \vdash A \text{ (on } \text{Suf}_A(\Phi_N)) \mid A \text{ is a type} \} \\ &\cup \{ A \vdash \Lambda \text{ (on } \Omega - \text{Nec}_A(\Phi_N)) \mid A \text{ is a type} \} \end{aligned}$$

where Λ is our notation for the empty set of types.

Back to the main discussion, we are assuming in this section an arbitrary set of axioms called *Givens* which contains all of the identity axioms $A \vdash A$ (on Φ). Our goal is still to tell the validity of a sequent $\Gamma \vdash \Delta$ (on Φ). Furthermore, we want to do it by resorting to a shrinking strategy like the one used by S^3 . The following relation intends to capture the general idea behind the process of shrinking the set of possible counterexamples as much as possible.

Definition 1 *Let $\text{Seq}, \text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_n$ be extended sequents. Then*

$$\text{Seq} \Rightarrow \text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_n$$

if the set of counterexamples to Seq is the union of the sets of counterexamples to $\text{Seq}_1, \text{Seq}_2, \dots, \text{Seq}_n$.

Proposition 1 *The following are properties of the \Rightarrow relation.*

1. *If $\text{Seq} \Rightarrow \text{Seq}_1$ then $\text{Seq}_1 \Rightarrow \text{Seq}$.*
2. *If $\text{Seq} \Rightarrow \text{Seq}_1, \dots, \text{Seq}_n, \text{Seq}^*$ and $\text{Seq}^* \Rightarrow \text{Seq}_{n+1}, \dots, \text{Seq}_{n+k}$, then*

$$\text{Seq} \Rightarrow \text{Seq}_1, \dots, \text{Seq}_n, \text{Seq}_{n+1}, \text{Seq}_{n+k}.$$

3. *If $\text{Seq} \Rightarrow \text{Seq}_1, \dots, \text{Seq}_n, \text{Seq}^*$, and Seq^* is a valid sequent then*

$$\text{Seq} \Rightarrow \text{Seq}_1, \dots, \text{Seq}_n.$$

4. *If*

$$\frac{\Gamma_1 \vdash \Delta_1 \text{ (on } \Psi), \dots, \Gamma_n \vdash \Delta_n \text{ (on } \Psi), \Gamma_* \vdash \Delta_* \text{ (on } \Psi)}{\Gamma \vdash \Delta \text{ (on } \Psi)}$$

is a sound rule which is either reversible or has $n = 0$, then it is also true that: If $\Gamma_ \vdash \Delta_*$ (on Ψ) is valid, then for any condition (i.e. basic set) Φ*

$$\Gamma \vdash \Delta \text{ (on } \Phi) \Rightarrow \Gamma_1 \vdash \Delta_1 \text{ (on } \Phi \cap \Psi), \dots, \Gamma_n \vdash \Delta_n \text{ (on } \Phi \cap \Psi), \Gamma \vdash \Delta \text{ (on } \Phi - \Psi)$$

The proof of this proposition is immediate from the definition of \Rightarrow . In the statement of property (4), which will play a key role in the definition of our reduction system, "reversible rule" means simply a rule which is bi-directional in the sense that if you know that the premises are valid then the conclusion

is valid, and if you know that the conclusion is valid, then all the premises are valid.

Now, we want our reduction system to be at least as powerful as the deduction system given by the three classical deduction rules. So following (4), we define three sound reduction rules, Substitution, Arrow-Weakening and Arrow-Cut, as follows. First, we have a Substitution Rule, which is simply property (2) above.

Substitution. If $Seq \Rightarrow Seq_1, \dots, Seq_n, Seq^*$ and $Seq^* \Rightarrow Seq_{n+1}, \dots, Seq_{n+k}$, then

$$Seq \Rightarrow Seq_1, \dots, Seq_n, Seq_{n+1}, Seq_{n+k}.$$

In order to obtain Arrow-Cut and Arrow-Weakening, we use a different form of the classical Cut rule which is equivalent to Standard Cut, but whose translation obtained by using property (4) above does give us the deduction power that we want⁹.

Definition 2 If Γ, Δ, S are finite sets of types, and Φ is a condition (i.e. a basic set), then

$$\Sigma(S, \Gamma, \Delta, \Phi) = \{\Gamma, \Gamma' \vdash \Delta, \Delta' \text{ (on } \Phi) \mid \Delta' \cap \Gamma' = \emptyset \text{ and } \Delta' \cup \Gamma' = S\}$$

is the set of weakenings of $\Gamma \vdash \Delta$ (on Φ) induced by S .

Proposition 2 Standard Cut rule is equivalent to the set of rules

$$(Cut_S) \quad \frac{\Sigma(S, \Gamma, \Delta, \Phi)}{\Gamma \vdash \Delta \text{ (on } \Phi)}, \quad \text{for } S \text{ finite}$$

Proof. Let $CUT = \{Cut_S \mid S \text{ is finite}\}$. To see that Standard Cut follows from CUT, it is enough to observe that

$$\frac{\Gamma, A \vdash \Delta \text{ (on } \Phi) \text{ and } \Gamma \vdash \Delta, A \text{ (on } \Phi)}{\Gamma \vdash \Delta \text{ (on } \Phi)}$$

is simply $Cut_{\{A\}}$. Now we need to prove that for each S finite, Cut_S follows from Standard Cut. We proceed by induction on $|S|$. The base case, $|S| = 0$, is trivial. Assume now that we have proven the result for every finite set of cardinality n . Let S be a set with $n + 1$ elements, say $S = T \cup \{A\}$, where $A \notin T$. We want to see that if we have proofs of each sequent in $\Sigma(S, \Gamma, \Delta, \Phi)$ we also have a proof of $\Gamma \vdash \Delta$ (on Φ). By the inductive hypothesis, it is enough to show that we have a proof for each sequent in $\Sigma(T, \Gamma, \Delta, \Phi)$. Then let $Seq = \Gamma, \Gamma' \vdash \Delta, \Delta'$ (on Φ) be a sequent such that $\Delta' \cap \Gamma' = \emptyset$ and $\Delta' \cup \Gamma' = T$, so that $Seq \in \Sigma(T, \Gamma, \Delta, \Phi)$. Hence, the two sequents

$$\Gamma, \Gamma', A \vdash \Delta, \Delta' \text{ (on } \Phi) \quad \text{and} \quad \Gamma, \Gamma' \vdash \Delta, \Delta', A \text{ (on } \Phi)$$

⁹There are counterexamples which show that the translation of the Standard Cut rule induce a weaker reduction system.

belong to $\Sigma(S, \Gamma, \Delta, \Phi)$, so they are provable, and from Standard Cut it follows that Seq is provable as well.

Our Arrow-Cut reduction rule (the “ \Rightarrow ” version of the Cut rule) is in fact a blend of the standard Cut and Weakening rules, and is given below.

Arrow-Cut. If there is an extended sequent $\Gamma^* \vdash \Delta^*$ (on Ψ) in the *Givens* such that $(\Gamma \cap \Gamma^*) \cup (\Delta \cap \Delta^*) \neq \emptyset$ then for any condition Φ ,

$$\Gamma \vdash \Delta \text{ (on } \Phi) \Rightarrow \{\Gamma \vdash \Delta \text{ (on } \Phi - \Psi)\} \cup (\Sigma(S, \Gamma, \Delta, \Phi \cap \Psi) - \{Seq\}).$$

where $S = (\Gamma^* - \Gamma) \cup (\Delta^* - \Delta)$ and $Seq = \Gamma, \Gamma^* \vdash \Delta, \Delta^*$ (on $\Phi \cap \Psi$).

There are two important observations regarding this reduction rule. First of all, the non-disjointness condition in Arrow-Cut is a relevance condition saying “look only at sequents in the *Givens* which are somehow related to $\Gamma \vdash \Delta$ (on Φ)”. The second observation is that Arrow-Weakening, the reduction rule corresponding to the standard Weakening rule, is just a particular case of Arrow-Cut, which occurs when the set S turns out to be the empty set of types. Just for completeness, we state the Arrow-Weakening reduction rule below.

Arrow-Weakening. If $\Gamma \vdash \Delta$ (on Ψ) is a weakening of a sequent $\Gamma^* \vdash \Delta^*$ (on Ψ) in the *Givens*, then for any condition Φ ,

$$\Gamma \vdash \Delta \text{ (on } \Phi) \Rightarrow \Gamma \vdash \Delta \text{ (on } \Phi - \Psi).$$

Derivations, as defined below, are the counterpart of standard proofs in standard Gentzen-style proof systems.

Definition 3 *Let Seq be a extended sequent and let SEQ be a set of extended sequents. A derivation of $Seq \Rightarrow SEQ$ from the Givens is a sequence*

$$\begin{aligned} Seq_0 &\Rightarrow SEQ_0 \\ Seq_1 &\Rightarrow SEQ_1 \\ &\vdots \\ Seq_n &\Rightarrow SEQ_n \end{aligned}$$

of instances of \Rightarrow such that:

1. $Seq_0 = Seq_n = Seq$ and $SEQ_n = SEQ$.
2. For each k , $0 < k \leq n$, $Seq_k = Seq_0$ or $Seq_k \in \bigcup_{0 \leq i < k} SEQ_i$.
3. For each k , $0 < k \leq n$, $Seq_k \Rightarrow SEQ_k$ is an instance of Arrow-Cut or $Seq_k \Rightarrow SEQ_k$ is obtained by Substitution from previous elements in the sequence.

In what follows, we will use the word *derivation* only in the sense of the previous Definition, while the word *proof* will always refer to proofs in Gentzen-style

systems with rules like Cut and Weakening. Here is an example of a derivation in our framework.

Example. Assume that the set of *Givens* is

$$Givens = \left\{ \begin{array}{l} A, B, C \vdash D \text{ (on } \Phi_0), \\ A, B \vdash C, D \text{ (on } \Phi_1), \\ A \vdash B, C, D \text{ (on } \Phi_2), \\ A, C \vdash B, D \text{ (on } \Phi_3) \end{array} \right\}$$

and we want to check for the validity of $A \vdash D$ (on Φ). Well, by Arrow-Cut with $S = \{B, C\}$ we have

$$A \vdash D \text{ (on } \Phi) \Rightarrow \left\{ \begin{array}{l} A \vdash D \text{ (on } \Phi \cap \overline{\Phi_0}), \\ A, B \vdash C, D \text{ (on } \Phi \cap \Phi_0), \\ A \vdash B, C, D \text{ (on } \Phi \cap \Phi_0), \\ A, C \vdash B, D \text{ (on } \Phi \cap \Phi_0) \end{array} \right\}$$

Applying Arrow-Weakening several times, we also get

$$A \vdash D \text{ (on } \Phi) \Rightarrow \left\{ \begin{array}{l} A \vdash D \text{ (on } \Phi \cap \overline{\Phi_0}), \\ A, B \vdash C, D \text{ (on } \Phi \cap \Phi_0 \cap \overline{\Phi_1}), \\ A \vdash B, C, D \text{ (on } \Phi \cap \Phi_0 \cap \overline{\Phi_2}), \\ A, C \vdash B, D \text{ (on } \Phi \cap \Phi_0 \cap \overline{\Phi_3}) \end{array} \right\}$$

Observe that the union of the conditions in the left is $\Phi - (\Phi \cap \Phi_0 \cap \Phi_1 \cap \Phi_2 \cap \Phi_3)$. Those are exactly the states for which a contradiction could be proven from Standard Cut when using a standard proof system.

The following proposition says that we can extract as much information from the *Givens* by using derivations as we can extract by using proofs based on Standard Cut, Identity and Weakening. More concretely, it follows immediately from this proposition that if you are trying to establish the validity of a sequent $\Gamma \vdash \Delta$ (on Φ) and this sequent can be proven from the *Givens* using a *standard* proof system, then our *reduction* system is guaranteed to shrink the set of possible counterexamples for $\Gamma \vdash \Delta$ (on Φ) from Φ to the empty set. That is, in this case the reduction system can prove that there are no counterexamples for the sequent.

Proposition 3 *Assume that from the Givens you can prove $\Gamma \vdash \Delta$ (on Ψ) using Standard Cut and Weakening. Then for every Φ there is a set of extended sequents $SEQ = \{Seq_1, \dots, Seq_n\}$ such that the union of the conditions that appear in SEQ is $\Phi - \Psi$ and $\Gamma \vdash \Delta$ (on Φ) \Rightarrow SEQ is derivable.*

In order to prove the Proposition, we prove the following lemma first.

Lemma 1 *Assume that from the Givens you can prove $\Gamma \vdash \Delta$ (on Φ) by using only Weakening and Standard Cut. Then from the Givens you can prove*

$\Gamma \vdash \Delta$ (on Φ) by using only Weakening and the rules Restricted Cut given by:

$$(Restricted\ Cut_{S,\Psi}) \quad \frac{\Sigma(S, \Gamma', \Delta', \Psi)}{\Gamma' \vdash \Delta' \text{ (on } \Psi)}$$

where there is a sequent $Seq \in \Sigma(S, \Gamma', \Delta', \Psi)$ which is a weakening of a sequent $\Gamma^* \vdash \Delta^*$ (on Ψ^*) in the *Givens* such that $S = (\Gamma^* - \Gamma') \cup (\Delta^* - \Delta')$ and $(\Gamma^* \cap \Gamma') \cup (\Delta^* \cap \Delta') \neq \emptyset$.

Proof of Lemma. Suppose that $\Gamma \vdash \Delta$ (on Φ) follows from $\Sigma(S, \Gamma, \Delta, \Phi)$ by Standard Cut, that all the sequents in $\Sigma(S, \Gamma, \Delta, \Phi)$ are provable by using only Weakening and Restricted Cut and that no sequent in $\Sigma(S, \Gamma, \Delta, \Phi)$ is a weakening of some sequent $\Gamma^* \vdash \Delta^*$ (on Φ^*) in the *Givens* such that $S = (\Gamma^* - \Gamma) \cup (\Delta^* - \Delta)$ and $(\Gamma^* \cap \Gamma) \cup (\Delta^* \cap \Delta) \neq \emptyset$.

- CASE 1: Suppose that there is a sequent $\Gamma^* \vdash \Delta^*$ (on Φ^*) in the *Givens* such that $(\Gamma^* \cap \Gamma) \cup (\Delta^* \cap \Delta) \neq \emptyset$, and let $S^* = \Gamma^* \cup \Delta^*$. Now, since each one of the sequents in $\Sigma(S, \Gamma, \Delta, \Phi)$ are provable by using only Weakening and Restricted Cut, each one of the sequents in $\Sigma(S \cup S^*, \Gamma, \Delta, \Phi)$ is also provable by using only Weakening and Restricted Cut (because only Weakening is needed to get proofs for sequents in $\Sigma(S \cup S^*, \Gamma, \Delta, \Phi)$ from proofs for sequents in $\Sigma(S, \Gamma, \Delta, \Phi)$). But now, $\Gamma \vdash \Delta$ (on Φ) can be proven from $\Sigma(S \cup S^*, \Gamma, \Delta, \Phi)$ by an application of Restricted Cut, and we are done.
- CASE 2: Suppose that there is no sequent $\Gamma^* \vdash \Delta^*$ (on Φ^*) in the *Givens* such that $(\Gamma^* \cap \Gamma) \cup (\Delta^* \cap \Delta) \neq \emptyset$. Then all the types in $\Gamma \cup \Delta$ have been introduced in the proofs of the sequents in $\Sigma(S, \Gamma, \Delta, \Phi)$ by Weakening. For each sequent $Seq \in \Sigma(S, \Gamma, \Delta, \Phi)$, let P_{Seq} be a proof for it. Modify P_{Seq} by deleting all the occurrences of elements in $\Gamma \cup \Delta$. Then we have that all the sequents in $\Sigma(S^*, \emptyset, \emptyset, \Phi)$ are provable. Hence $\Lambda \vdash \Lambda$ (on Φ) follows from Cut, and now by applying Weakening we can also prove $\Gamma \vdash \Delta$ (on Φ).

Proof of Proposition. By the above lemma it is enough to prove that if from the *Givens* you can derive $\Gamma \vdash \Delta$ (on Ψ) by using Restricted Cut and Weakening, then there exists a set of extended sequents $SEQ = \{Seq_1, \dots, Seq_n\}$ such that the union of the conditions in SEQ is $\Phi - \Psi$ and $\Gamma \vdash \Delta$ (on Φ) \Rightarrow SEQ is derivable. We proceed by induction on proofs. Let P be a proof for $\Gamma \vdash \Delta$ (on Ψ) from the *Givens* that uses only Weakening and Restricted Cut.

- The base case is $|P| = 1$. If $\Gamma \vdash \Delta$ (on Ψ) belongs to the set of *Givens*. Then

$$\Gamma \vdash \Delta \text{ (on } \Phi) \Rightarrow \{\Gamma \vdash \Delta \text{ (on } \Phi - \Psi)\},$$

is just an instance of Weakening.

- Now assume that $\Gamma \vdash \Delta$ (on Ψ) follows from $\Gamma' \vdash \Delta'$ (on Ψ) by Weakening, that is, $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$. Moreover, assume that we have a proof for

$\Gamma' \vdash \Delta'$ (on Ψ). Then by induction hypothesis there is a set SEQ' such that

$$\Gamma' \vdash \Delta' \text{ (on } \Phi) \Rightarrow SEQ'$$

is derivable (by a derivation \mathfrak{D} , say) and the union of conditions in SEQ' is $\Phi - \Psi$. Observe that then we obtain a derivation of

$$\Gamma \vdash \Delta \text{ (on } \Phi) \Rightarrow SEQ'$$

by modifying \mathfrak{D} slightly. Namely, replace each sequent $\Gamma_1 \vdash \Delta_1$ (on Ψ') that appears in \mathfrak{D} by $\Gamma_1 \cup \Gamma \vdash \Delta_1 \cup \Delta$ (on Ψ').

- Finally, assume that $\Gamma \vdash \Delta$ (on Ψ) follows from $\Sigma(S, \Gamma, \Delta, \Psi)$ by Restricted Cut. Let $\Sigma(S, \Gamma, \Delta, \Psi) = \{Seq_1 \text{ (on } \Psi), \dots, Seq_n \text{ (on } \Psi), Seq^* \text{ (on } \Psi)\}$ and using the inductive hypothesis suppose that for each Seq_i ($i = 1, \dots, n$) with $Seq_i = \Gamma, \Gamma_i \vdash \Delta, \Delta_i$, there is a set SEQ_i and a derivation \mathfrak{D}_i of $Seq_i \text{ (on } \Psi) \Rightarrow SEQ_i$ such that the union of conditions in SEQ_i is the empty set. Assume also that $Seq^* \text{ (on } \Psi)$ is a weakening of a sequent $\Gamma^* \vdash \Delta^* \text{ (on } \Psi^*)$ which belongs to the *Givens* and such that $(\Gamma^* \cap \Gamma) \cup (\Delta^* \cap \Delta) \neq \emptyset$ and $S = (\Gamma^* - \Gamma) \cup (\Delta^* - \Delta)$. Let $\Sigma_S = \{Seq_1 \text{ (on } \Psi), \dots, Seq_n \text{ (on } \Psi)\}$. Then,

$$\begin{aligned} \Gamma \vdash \Delta \text{ (on } \Phi) &\Rightarrow \{\Gamma \vdash \Delta \text{ (on } \Phi - \Psi)\} \cup \Sigma_S \\ &\quad \mathfrak{D}_0 \\ &\quad \vdots \\ &\quad \mathfrak{D}_n \\ \Gamma \vdash \Delta \text{ (on } \Phi) &\Rightarrow \{\Gamma \vdash \Delta \text{ (on } \Phi - \Psi)\} \cup (\Sigma_S - \{Seq_0 \text{ (on } \Psi)\}) \cup SEQ_0 \\ \Gamma \vdash \Delta \text{ (on } \Phi) &\Rightarrow \{\Gamma \vdash \Delta \text{ (on } \Phi - \Psi)\} \cup (\Sigma_S - \{Seq_0 \text{ (on } \Psi), Seq_1 \text{ (on } \Psi)\}) \\ &\quad \cup SEQ_0 \cup SEQ_1 \\ &\quad \vdots \\ \Gamma \vdash \Delta \text{ (on } \Phi) &\Rightarrow \{\Gamma \vdash \Delta \text{ (on } \Phi - \Psi)\} \cup \bigcup_{i \leq n} SEQ_i \end{aligned}$$

is a derivation such that the union of conditions in the right is $\Phi - \Psi$, as desired. The first step in the derivation is an instance of Arrow-Cut, and the last n steps are instances of Substitution.

0.5 Conclusions and Future Work

We have described S^3 , an inference engine implemented in *Mathematica*, which performs inference based on the existence of knowledge at two levels. The pre-symbolic level is modeled by means of a state space and the symbolic level by a logical language of (situation) types. In principle, there is no perfect translation between representations from one level to equivalent representations in the other level. Instead, both levels interact and cooperate in order to draw conclusions.

In addition, S^3 features inference which is context dependent, non-monotonic phenomena and a possible model of attentional effects in inferences performed by cognitive agents. We have presented also a first step towards a formalization and generalization of the inference process used by S^3 by means of reduction rules. The reduction rules capture the strategy used by S^3 , namely, trying to shrink the set of possible counterexamples for a sequent as much as possible.

Our formalization generalizes the reduction process in S^3 in the sense that it provides a way to use sequents which have been already proved to be valid, while in S^3 that kind of memory does not exist currently. However, the whole dynamics of the process of iterative shrinking performed by S^3 when solving queries, which involves updating the necessary and sufficient conditions to be used at each step, is not captured yet by our reduction rules system. This is just one of the various things that need to be solved in the future. As an example, another direction of improvement has to do with the use of additional structure of the state space in order to guide different parts of the process, especially the search for counterexamples which has to be done in cases where the shrinking process lead by reduction rules has finished without allowing a decision about the validity of a sequent.

A lot of work has been done in areas that overlap with our project, especially heterogeneous reasoning (as in [?]) and commonsense reasoning (as in [?], [?], [?]).

Bibliography

- [AB 96] Allwein, G. and Barwise, J. eds., “Logical Reasoning with Diagrams”. Oxford University Press, 1996.
- [ABTY 01] Akman, V., Bouquet, P., Thomason, R., Young R.A. eds., “Modeling and Using Context”. Third International and Interdisciplinary Conference, CONTEXT, 2001, Dundee, UK, July 27-30, 2001. Proceedings. Lecture Notes in Artificial Intelligence 2116, Springer Verlag.
- [A 97] Akman, V. and Surav, M., “The Use of Situation Theory in Context Modeling”. *Computational Intelligence* 13 (3), August 1997; 427-438.
- [B 97] Barwise, J., “Information and Impossibilities”. *Notre Dame Journal of Formal Logic* 38 (4), Fall 1997; 488-515.
- [BS 97] Barwise, J. and Seligman, J. “Information Flow: The Logic of Distributed Systems”, Cambridge Tracts in Theoretical Computer Science 44, Cambridge University Press, 1997.
- [BP 83] Barwise, J. and Perry, J. “Situations and Attitudes”. MIT Press, 1983.
- [C 92] Casti, J., “Reality Rules”, I. Wiley Interscience, New York, 1992.
- [G 00] Gardenfors, P., “Conceptual Spaces: The Geometry of Thought”, MIT Press, 2000.
- [TA 97] Tin, E. and Akman, V., “Situated Non-monotonic Temporal Reasoning with BABY-SIT”. *AI Communications* 10 (2), July 1997; 93-109.