1 Introduction

Humans can at least apparently conduct deductive reasoning. Various logical inference systems have been developed in order to characterize such human reasoning. The strength of such a properly logical analysis of human reasoning lies in its productivity and predictive power; it enables the theorist to define precisely which data-space the theory covers and then mathematically prove that the theory covers the intended data-space sound and complete. However, as a demerit of a properly logical approach to human reasoning, it is not clear how humans have developed the ability to deal with the knowledge representations that such an analysis indicates. The structure of the logical language representations, which includes abstract elements such as quantifiers and disjunction operators, deviates greatly from the structure of the representations that the other cognitive mechanisms (such as vision and auditory perception) deal with. Thus, a properly logical analysis of human reasoning implies a larger evolutionary gap in the development of this cognitive ability. Moreover, the presence of quantifiers and other propositional operators in the logical language representation makes it harder to ground logical inferences in perception, since perceptual representations do not seem to include corresponding elements. The larger discrepancy between the logical and non-logical representations as above also poses a problem for the computational theory of human-level intelligence. Such a computational theory typically uses both logical and non-logical methods depending on the task, and each of these computational methods uses a different kind of representational structure. In order to improve the efficiency of the spontaneous data-exchange between the logical and non-logical computation methods, it is better to make the knowledge representation that
the logical method uses structurally closer to the representations that non-logical methods use, such as topological space for visual computation and some sort of spectrogram-based data-structure for auditory information processing.

Enabling general reasoning in terms of simulations conducted with perceptual mechanisms can solve the above problems, since the language representation that perceptual simulations use does not include abstract elements such as quantifiers and negative and disjunctive operators. The crucial question is how well such an approach can maintain the productivity and predictive power of a properly logical approach. My Issues paper in 2011 has shown that our simulation language can be at least as expressive as classic first-order logic. However, since the paper did not provide the precise inference rules and their intended interpretations in some well-defined semantic structure, such as a finitely axiomatizable algebra, it is not clear if the theory of reasoning using our simulation language is as predictive as the one using a proof theoretically well-behaved logical inference system, such as the sequent calculus for FOL. Admittedly, analyzing the deductive reasoning that humans conduct in terms of perceptual simulations indicates that the deductive reasoning needs not be enabled in terms of a properly logical inference system with its expected properties such as consistency, decidability and completeness. However, giving up these proof theoretic properties means that the resultant theory of human reasoning cannot strictly predict either its precise intended data coverage or whether the theory can actually achieve the intended data coverage in a complete manner. This is because the inference patterns that humans’ deductive reasoning covers is potentially infinite and at least apparently, humans seem to be achieving this data coverage by way of a finite number of inference rules.

Based on these considerations, this talk gives the first stab at this problem of axiomatizing the inference rules using our simulation language, as well as speculating what intended semantic structure this inference system would be sound and complete to. Since this is just an initial attempt, I do this by comparing our putative inference system with the sequent calculus for classic first-order logic, which arguably has the best proof theoretical properties among its established competitors.

Section 2 explains our simulation language. Section 3 discusses the inference rules using this language, in comparison to the inference rules in the Gentzen sequent calculus for standard first-order logic (FOL).
2 Simulation Language

2.1 Review of our previous work

This subsection summarizes the simulation language that we presented in Uchida, Cassimatis, and Scally (2012), before we modify it into a language that is more suitable for our ultimate goal.

In the above paper, we presented a formal language that closely reflects the cognitive simulation process that humans are assumed to go through. For example, consider (1).

(1) In Reality: \{\text{NameOf}(j, \text{John}), \text{NameOf}(m, \text{Meg}), \ldots\}

Simulation 1, based on (the assumptions): \{\text{Run}(j)\} / Conclusion: \{\text{Run}(meg)\}

The set followed by ‘In Reality’ in (1) is called the reality set. It represents the knowledge that the agent (i.e., the person who runs mental simulations) has about the real world. Simulation 1 in (1) then means that the mental simulation based on the assumption that the particular individual \(j\) in the real world (whose name is, as the reality set indicates, John) leads to the conclusion that another particular individual in the real world, named Meg, runs. This simulation corresponds to the proposition, ‘If John runs, Meg runs.’ Crucially, in the above paper, we argued that the assumption-conclusion structure\(^1\) that we posited at (1) is reflected in the representations that the perceptual mechanisms, such as visual information processing, deal with.

We show our treatment of universal quantifier in (2).

(2) Simulation 2, based on: \{\text{Dog}(d), \text{New}(d)\} / Conclusion: \{\text{Bark}(d)\}

Simulation 2 in (2) means that if the agent posits an imaginary individual \(d\) in his mental simulation and assumes that \(d\) is a dog (and the agent attributes no other properties or relations to this imaginary individual), and if this agent somehow can definitely conclude that this object \(d\) barks based only on this assumption, then it means that this agent somehow believes that every dog barks (since otherwise, the agent would not be able to

---

\(^1\)As we did in (1), this paper focuses only on the simulation in which the assumption to the left of the slash, ‘/’, (together with the person who runs this simulation knows about the real world, see below) entails the conclusion to the right of ‘/’. In a complete formulation, we can add ‘weights’ to simulations, in order to indicate with how much certainty the conclusion follows from the assumption. We use weights in this way mostly for dealing with the uncertainty of human reasoning, but we ignore this extra complexity in this paper. We sometimes call a simulation such as Simulation 1 in (1) a hard constraint, since the conclusion follows from the assumption.
conclude that $d$ barks, only based on the assumption that $d$ is a dog and nothing else. The other atomic proposition $New(d)$ in the assumption of Simulation 2 is a syntactic sugar that indicates that the individual $d$ does not appear anywhere else in this agent’s knowledge base (or in any other simulations that this agent previously ran). This makes sure that the agent does not associates any other properties than the property DOG with this imaginary individual, which in turn ensures that the simulation conclusion, $Bark(d)$, depends only on the assumption that this individual is a dog. Now the crucial point about the simulation language that, given that the simulation that we have just sketched unambiguously indicates the presence of the universally quantified belief ‘Every dog barks,’ we can let the symbolic representation in (2) represent this universally quantified proposition. Note that we are not confusing the simulation above and the universally quantified belief that leads to that simulation result. Since the simulation result above requires the universal belief ‘Every dog barks,’ we can use the symbolic representation of this simulation to unambiguously indicate the presence of this universally quantified proposition. The obvious benefit of this approach is that we now do not need an explicit quantifier in our symbolic representation in order to express a universally quantified proposition. This makes our symbolic simulation language better grounded in perception than a language that explicitly represents a quantifier, such as FOL.

Below, we enumerate examples of how our simulation language as above can handle various elements of FOL without explicitly specifying abstract operators and connectives. We add some brief comments after each example. For the support of the simulation mechanisms that we use, see Uchida, Cassimatis, and Scally (2012). In each example, a. shows the natural language proposition and the FOL formula for it and b. shows how the simulation language represents that proposition.

(3) Existential quantifier:

a. John speaks a language. $\exists x (Language(x) \land Speak(john, x))$

b. In Reality: \{NameOf(j, ‘John’)\},

S3, based on: \{} / Conclusion: \{New(l), Language(l), Speak(j, l)\}

S(imulation) 3 in (3-b) means that if the agent simulates with regard to the particular individual whose name the agent knows is ‘John’, then this simulation concludes that this individual $j$ speaks a language. The other object, $l$, is introduced newly in the conclusion, as the syntactic sugar $New(l)$ indicates, and the rule of thumb here is that if some new object is introduced in the conclusion of a simulation, it implies existential quantification.
with regard to that object. For detailed justification for this assumption, again, see Uchida, Cassimatis, and Scally (2012), but just suppose that an agent is imagining an encounter with a new (imaginary) friend. That new friend will soon become a particular individual, while the agent attributes more and more properties to that individual as the reasoning proceeds further, but at the moment of her introduction of a new friend into his imaginary world, the correct description would be something like ‘I (= the agent) am meeting a friend’, rather than, ‘I am meeting Meg (assuming that this new friend’s name later turns out to be Meg)’ or ‘I am meeting the friend.’ In other words, introducing a new object into some sort of mental model amounts to existentially quantifying over the argument position that this object occupies.\(^2\) Again, since the simulation at (3-b) unambiguously implies the existential quantification with regard to ‘I’, we can use the symbolic representation of this simulation, that is S3 in (3), to express the corresponding existentially quantified proposition, ‘John speaks a language.’

To summarize our treatment of our quantification, if a new object is introduced in the assumption, or as we call the ‘basis’ of a simulation, then it implies universal quantification, as we saw at (2). If a new object in turn is introduced in the conclusion of a simulation, then it implies an existential quantification. For our treatment of scope dependency between quantifiers, see Uchida, Cassimatis, and Scally (2012).

Another important thing about S3 in (3-b) is that this simulation is not based on any particular assumption (that is, the set to the left of ‘/’ is empty). As we explained in details in Uchida, Cassimatis, and Scally (2012), this does not mean that the simulation S3 is not dependent on anything. Our simulation in this regard is that each simulation inherits all the knowledge that the agent has about the states of affairs in the real world (that is, formally, the content of the reality set, ‘In reality, {...}). Less formally, this means that (in the agent’s mind), everything that the agent knows is true in the real world is also true in each simulation that he runs (unless it is explicitly specified otherwise, as we see later in our treatment of ‘counterfactual’ reasoning).

Although every proposition in the set, ‘In reality: {...} is inherited by each simulation (again, unless specified otherwise), the propositions in the reality set are still not the basis (or the assumption) of each simulation. This is because the ‘fact’ in the real world is not an assumption (which is a hypothesis, not a fact). Thus, we can still say that Simulation 3 in (3-b) is not based on any assumptions. It only means that all the facts that the agent knows about the real world contributes to the agent’s deriving the conclusion of

\(^2\)This observation has been accommodated in other theories of human reasoning, such as Discourse Representation Theory, Kamp, van Genabith, and Reyle (2005).
this simulation, such as the kinds of factual information that the agent has gained via her acquaintance with this individual $j$ in the real world.

So far, the only elements that can be arguably taken to be ‘abstract’ are the introduction of ‘imaginary’ (or ‘new’) objects into the mental simulation and the assumption-conclusion structure, reflected by the slash-bar ‘/’ in notation. Both of these elements can actually be observed in the perceptual cognitive mechanisms, such as visual information processing by humans (cf. Hesslow (2002)), although for formal and computational reasons, we remove the latter from our language representation.

At this stage, however, we introduce into our simulation language an element that can be taken to be a more significant ‘jump’ from the other cognitive mechanisms such as visual and auditory information processing. Consider our treatment of negation at (4).

(4) a. John does not speak a language. $\neg \exists x. (\text{Language}(x) \land \text{Speak}(\text{john}, x))$.
   
   b. In reality, $\{\text{NameOf}(j, \text{John}), \ldots\}$,
      
      S5, based on: $\{\text{New}(l), \text{Language}(l), \text{Speak}(\text{john}, l)\}$ / Conclusion: $\{\bot\}$

The simulation S4 in (4-b) means that when the agent makes this simulation with regard to a particular person $j$, who she knows is named ‘John’, based on the assumption that the individual $j$ speaks some language, that simulation definitely fails. Since the new object ‘$l$’ in the basis of a simulation implies universal quantification, in a more logical description, (4-b) means that for all object $x$, if the agent assumes that John in the real world speaks $x$ and runs a mental simulation based on that assumption, then the simulation fails (in our notation, $\bot$ in the conclusion of a simulation means that the simulation fails). Now, consider what a simulation based on the assumption: $\{A_1, \ldots, A_n\}$ means from a cognitive viewpoint.

Naturally, such a simulation fails because the agent somehow understands that some of the assumptions $A_1, \ldots, A_n$ does not hold true. In (4-b), this has the effect of negating the conjunction of the formulas in the atom, that is, $\neg(\text{Language}(l) \land \text{Speak}(\text{john}, l))$ (note that $\text{New}(l)$ is just a syntactic sugar, in order to efficiently indicate that the object $l$ is introduced newly there, and so while this still has the effect of universally quantifying over the argument slot occupied by the term $l$, this formula does not get into the scope of negation). Based on such reasoning, S5 in (4-b) implies that for all $x$, it is not the case that $x$ is a language and John speaks $x$, or $\forall x((\text{Language}(x) \land \text{Speak}(j, x)) \rightarrow \bot)$ in FOL. Note that this FOL formula is equivalent to the formula in (4-a).

---

3We use this notation since when a simulation based on the assumption: $\{A_1, \ldots, A_n\}$ fails, it has the same semantics with the FOL formula ($\phi \rightarrow \bot$), which is equivalent to $\neg \phi$, where $\phi$ is the formula that we can get by conjoining the assumptions, $A_1, \ldots, A_n$.)
Now, although S5 in (4-b) has the same truth condition as the FOL formula in (4-a) as above, its form corresponds to the universally quantified formula as we showed above. Again, these two FOL formulas are logically equivalent and in that sense, (4-b) does its most fundamental job, but if we introduce another independently motivated element to our simulation language, we can more directly reflect the structure of the FOL formula in (4-a) (that is, except for the structure with regard to the explicit quantifiers). Consider (5).

(5) In Reality: \{NameOf(j, John)\},
    S3, based on: {} / Conclusion: \{New(l), Language(l), Speak(j, l)\}
    S6, based on: \{Hold(S3)\} / Conclusion: \{⊥\}

In (5), we have first repeated the simulation S3 in (3-b), together with the same reality set. Remember S3 with that reality set implies, ‘John speaks a language’ or \(\exists x (Language(x) \land Speak(john, x))\). Now the new simulation, S6, takes this simulation S3 as its assumption. In other words, S6 assumes that S3 holds.\(^4\) This new simulation S6 fails in (5). As we explained above, this has the effect of negating the content of the assumption, that is, \(Hold(S3)\). Since \(Hold(S3)\) implies \(\exists x (Language(x) \land Speak(john, x))\) (= ‘John speaks a language’), this means that S6 implies the negation of this formula, \(\neg \exists x (Language(x) \land Speak(john, x))\) (= ‘John does not speak a language’), which is (4-a). Thus, (5) directly reflects the structure of the FOL formula in (4-a) (while being logically equivalent to the simulation in (4-b)).

As far as we are aware, we have no evidence that the perceptual mechanism can run a simulation that is based on the assumption that another simulation holds. However, once the simulation language can introduce new objects (or events\(^5\)), it is not such a stretch to introduce a term such as S1 which refers back to another possible simulation and assumes its success (or failure), although there is still some non-trivial difference between simulating with a hypothetical event on the one hand and simulating based on the success of another

---

\(^4\)This means that if the agent assumes that a simulation holds in the basis of another simulation, it indicates that the agent is no longer sure whether the former simulation holds or not. For example, in (3), we assumed that the agent is certain that S3 holds, and hence, the agent believes the existentially quantified proposition in (3-b). In contrast, in (5), the agent only ‘assumes’ that S3 holds, that is, ‘\(Hold(S3)\)’ is just an assumption there, no longer a statement considered to be true, in the agent’s mind.

\(^5\)In a more precise formulation, we develop a simulation language that corresponds to an event theoretic first-order logic language, such as \(\exists e [Run(e) \land AgentOf(e, john)]\), instead of the non-event theoretic FOL formula, \(Run(john)\). Such simulations can hypothetically simulate a possible event, introducing a new event term into the simulation.
possible simulation, in that only the latter is reflexive in a sense.\textsuperscript{6}

Finally, once one simulation can be based on (the validity of) another simulation, our simulation language can represent the semantics of ‘OR’ as well.

(6)  

\begin{enumerate}
  \item John runs or Meg runs. \((\text{Run}(john) \lor \text{Run}(meg))\)
  \item In Reality: \{\text{NameOf}(j, John), \text{NameOf}(m, Meg), \ldots\},
    \begin{align*}
      & \text{S7, based on: } \{\text{Run}(j)\} / \text{Con: } \{\bot\}, \\
      & \text{S8, based on: } \{\text{Run}(m)\} / \text{Con: } \{\bot\}, \\
      & \text{S9, based on: } \{\text{Hold}(S7), \text{Hold}(S8)\} / \text{Con: } \{\bot\}
    \end{align*}
\end{enumerate}

S7, which is based on the assumption ‘John runs’ \((\text{=Run}(j))\) fails, and thus, S7 implies ‘John does not run’ \((\text{=¬Run}(john))\). Similarly, S8 implies ‘Meg does not run’ \((\text{=¬Run}(meg))\). Now, S9 is based on the assumption that both S7 and S8 holds, which means that S9 is based on the assumption that ‘John does not run and Meg does not run’ \((\text{=¬(¬Run}(john) \land \text{¬Run}(meg)))\). Since this simulation S9 fails, it implies the negation of the assumption above, that is ‘It is not the case that John does not run and Meg does not run’ \((\text{=¬(¬Run}(john) \land \text{¬Run}(meg)))\), which is logically equivalent to ‘John runs or Meg runs’ \((\text{= (Run}(john) \lor \text{Run}(john)).\text{)}\)

Our simulation treatment of disjunction at (20-b) is arguably indirect and complex. We can argue that the processing difficulty humans experience with disjunctive propositions support this analysis. However, in Section ??, we show another way of representing disjunctive propositions.

To summarize, each simulation is in the form of ‘Simulation X, based on: \{\text{A1, ..., Am}\} / \text{Con: } \{\text{C1, ..., Cn}\}, where \{\text{A1, ..., Am}\} is a possibly empty, finite set of assumptions and \{\text{C1, ..., Cn}\} is a non-empty, finite set of conclusions. Neither the assumptions nor the conclusions include any sort of propositional operators or connectives. Commas inside the both sets mean ‘AND’, as it is standard with the set theoretic notation. Since this paper only discusses ‘fully certain’ simulations, the relation between the assumptions to the conclusions corresponds to the semantics of the material implication, \(\rightarrow\), in classical logic.

\textsuperscript{6}We can speculate that the ability of a cognitive mechanism to reflexively hypothesize and evaluate what it can conduct is indeed an ability that separates primates and human beings. For example, a higher-order attitude attribution (the so-called ‘higher-order explicature’, cf. Sperber and Wilson (1995)) is something that only humans older than 5 seem to conduct (i.e., theory of mind) and there is a clear similarity between such a propositional attitude attribution and the attribution of the property ‘Hold’ to a simulation. Also, the agent’s ability to base one simulation on the success of another simulation as above would produce a potentially infinite embedding structure between simulations, and that can again be claimed to be a crucial difference between the cognitive ability of human beings and the cognitive ability of primates.
that is, the assumptions entail the conclusions. This simulation language can introduce new, hypothetical objects into simulations, represented with new argument terms. If a new term is introduced in the assumption (= basis) of a simulation, it implies a universal quantification. If a new term is introduced in the conclusion of a simulation, it implies an existential quantification. Everything that the agent knows is true in the real world (which includes everything that the agent has perceived to be true in the real world) is contained in the reality set in the form of ‘In Reality: {.....}.’ The contents of the reality set do not need to be fully specified (typically, they are only partially specified), but whatever is included in that set are assumed to be also true in each simulation that the same agent runs, unless otherwise specified. Finally, a simulation can depend on another simulation, and in the simulation language, this is achieved by including atomic propositions such as ‘Hold(Si)’ into a simulation Sj, where Si and Sj are different simulations.

Now, although the simulation language that we sketched in this section is a decent representation of the mental simulation that humans are assumed to conduct, this language is still not ideal for our overall goal, which includes the improvement of the efficiency of spontaneously switching between ‘logical’ and ‘non-logical’ data-structures depending on the purpose and the removal from the language representation any element that is not reflected in first-order interpretation model structures. More specifically, the slash connective ‘/’, whose intended interpretation is the same as ‘→’ in classical logic, compromises the efficiency of switching between logical and non-logical computational mechanisms and it also does not have a corresponding model theoretic object in the first-order model structures. We resolve this issue in the next section.

2.2 Improved Simulation Language

In order to improve the efficient communication between logical and non-logical computation methods and also the correspondence between the structures of simulation language and the model structures into which this language is interpreted, we remove ‘/’ from our simulation language. We also give up the multiple-set structure that we used in the previous section and express every proposition in one-set structure. In other words, the entire knowledge of the agent, which includes both her knowledge about the real world and all the simulations that she runs, is represented in one set of operator-free predicate-logic-like formulas. For example, Simulation 2 in (2) in the previous section, repeated here in (7-a), is represented as in(7-b).
In (7-b), we do not represent the complete knowledge and information that the agent has in her brain, as the commas ‘...’ in (7-b) indicate. In order to remove ‘/’, we introduce a new kind of argument terms to the atomic predicate-argument structures, that is, the argument terms such as ‘$w1b$’ and ‘$w1$,’ which specify where the formulas appear in the simulations. For example, ‘$\text{New}(d, w1b)$’ means that a new object $d$ is introduced in the basis of the simulation named $w1$ and ‘$\text{Dog}(d, w1b)$’ means that this new object is assumed to be a dog in the simulation $w1$. ‘$\text{Bark}(d, w1c)$’ means that this simulation $w1$ concludes that the object $d$ barks. By way of the reasoning that we explained with Simulation 2 in the previous section, the set in (7-b) implies, ‘Every dog barks’, just like S2 in (7-a). Note that for one simulation $w1$, we introduce three terms, $w1b$ (for the basis/assumption) and $w1c$ (for the conclusion), as well as $w1$, which refers to the entire simulation. The final argument of each formula always indicates where the formula is in the simulation indicated by the number index. Since each formula is either in the basis or in the conclusion of a simulation, the final argument is either $w1b$ or $w1c$. The argument $w1$ is used to refer back to this simulation in another simulation, as we saw in the previous section and as we see below as well.

We show how this improved simulation language deals with (3)$\sim$(20-b) in the previous section below, repeating the English and the corresponding simulations in the previous language. Everything that we explained about simulations in the previous section still holds. The only difference is the formal language which we use to represent those simulations.

(8) Existential quantifier:

a. John speaks a language. $\exists x (\text{Language}(x) \land \text{Speak}(j, x))$

b. In Reality: \{\text{NameOf}(j, John)\},
   S3, based on: \{\} / Conclusion: \{\text{New}(l), \text{Language}(l), \text{Speak}(j, l)\}

c. \{\text{NameOf}(j, John, R), \text{New}(l, w3c), \text{Language}(l, w3c), \text{Speak}(j, l, w3c), \ldots\}

The formula $\text{NameOf}(j, John, R)$ in (8-c) has $R$ as its final argument, which again is the location argument. Which means that this formula represents what the agent knows about the real world, that is, a formula in the reality set in the previous language in (8-b). The world $w3$ corresponds to Simulation 3 above. Since no formulas has ‘$w3b$’ as its final argument (which again is the ‘location’ argument that shows where the formula appears),
the basis of the simulation $w_3$ is empty, as with $S_3$ in (8-b).

(9)  
   a. John does not speak a language. $\neg\exists x. (\text{Language}(x) \land \text{Speak}(\text{john}, x))$.
   b. In reality, $\{\text{NameOf}(j, \text{John}), ...\}$,
      $S_5$, based on: $\{\text{New}(l), \text{Language}(l), \text{Speak}(\text{john}, l)\}$ / Conclusion: $\{\bot\}$
   c. $\{\text{NameOf}(j, \text{John}, R), \text{New}(l, w_5b), \text{Language}(l, w_5b), \text{Speak}(\text{john}, l, w_5b),$
      $\bot(w_5c)\}$

The simulation $w_5$ corresponds to $S_5$ and thus, all the formulas that have $w_5b$ as its last
argument are understood to be part of the assumptions of this simulation, whereas $\bot(w_5c)$
means that $\bot$ is in the conclusion of the simulation $w_5$, indicating that the simulation $w_5$
fails, as we explained in the previous section.

The rest of the examples (4)$\sim$(20-b) in the previous sections can be handled in a similar
manner. We pair each of these in a. with the corresponding simulation expression in the
new language in b.

(10)  
   a. In Reality: $\{\text{NameOf}(j, \text{John})\}$,
      $S_3$, based on: $\{}$/ Conclusion: $\{\text{New}(l), \text{Language}(l), \text{Speak}(j, l)\}$
      $S_6$, based on: $\{\text{Hold}(S_3)\}$ / Conclusion: $\{\bot\}$
   b. $\{\text{NameOf}(j, \text{John}, R), \text{New}(l, w_3c), \text{Language}(l, w_3c), \text{Speak}(j, l, w_3c),$ $\text{Hold}(w_3, w_6b), \bot(w_6c), ...\}$

In (10-b), $\text{Hold}(w_3, w_6b)$ means that this formula appears in the basis of the simulation $w_6$,
which means that the simulation $w_3$ is assumed to hold in the assumption of the simulation
$w_6$. The rest of the explanation is the same as the explanation for (5), and (10-b) represents
Jack does not speak any language’ ($= \neg\exists x. (\text{Language}(x) \land \text{Speak}(\text{john}, x))$).

Finally, we show how the new simulation language handles a disjunction.

(11)  
   a. John runs or Meg runs. $(\text{Run}(\text{john}) \lor \text{Run}(\text{meg}))$
   b. In Reality: $\{\text{NameOf}(j, \text{John}), \text{NameOf}(m, \text{Meg}), ...\}$,
      $S_7$, based on: $\{\text{Run}(j)\}$ / Con: $\{\bot\}$
      $S_8$, based on: $\{\text{Run}(m)\}$ / Con: $\{\bot\}$,
      $S_9$, based on: $\{\text{Hold}(S_7), \text{Hold}(S_8)\}$ / Con: $\{\bot\}$
   c. $\{\text{NameOf}(j, \text{John}, R), \text{NameOf}(m, \text{Meg}, R), \text{Run}(j, w_7b), \bot(w_7c),$ $\text{Run}(m, w_8b), \bot(w_8c), \text{Hold}(w_7, w_9b), \text{Hold}(w_8, w_9b), \bot(w_9c), ...\}$
Again, the simulation \( w_7 (= S_7) \) in (11-c) means \( \neg \text{Run}(john) \) and the simulation \( w_8 (= S_8) \) there means \( \neg \text{Run}(meg) \), since the simulation \( w_9 (= S_9) \), which takes \( w_7 \) and \( w_8 \) as the assumptions, fail, \( w_9 \) implies the negation of the above two formulas, that is, \( \neg(\neg \text{Run}(john) \land \neg \text{Run}(meg)) \), equivalent to \( (\text{Run}(john) \lor \text{Run}(meg)) \).

We assume that the set such as the one in (11-c) increases its contents as the agent acquires more knowledge about the real world and as the agent runs more mental simulations.

To summarize, in this improved simulation language, every FOL formula can be expressed in one set of operator-free formulas. Each formula in this set has the form,

\[ \text{Predicate}(t_1, ..., t_n, w), \]

where the final argument \( w \) is the location term (which we call the world argument), indicating which simulation the formula appears in and whether it is in the basis or in the conclusion of that simulation. If the world argument is \( R \), it means that the formula is part of what the agent knows is true in the real world. The other world arguments \( w_1, ..., w_n \) represent simulations, but since each formula appears either in the basis or in the conclusion of a simulation, the final argument of each formula is either \( wib \) or \( wic \), where \( i \in \mathbb{N} \). Like the simulation language in the previous section, this new language can also allow us to refer to a simulation in another simulation, such as one simulation is based on the assumption that another simulation holds (or fails). In this way, this language can also express the propositions that are represented by FOL formulas that involve complex embedding structures using \( \lor \) (OR) and negation, \( \neg \).

### 3 Inference rules

This section considers a possible set of inference rules that we can provide with our simulation language. Ultimately, a finite axiomatization of the inference rules paired with a well-defined algebraic structure being its intended semantics will maximize the predictive power of our theory of reasoning, which in turn will show that our simulation language can actually replace a standard logical language without compromising its main strength. However, this paper only provides an initial attempt, which turns out to be falling short of finitely axiomatizing the inference rules. But our attempt also shows that simulations can capture most of the sequent calculus inference patterns with the intended semantics and the remaining discrepancy is at least apparently not fatal. This indicates that we should be able to finitely axiomatize our inference rules in our future research.

First, we show some example inferences using two major proof representation systems,
the Natural Deduction calculus for FOL and the Gentzen sequent calculus for FOL. We then provide our own inference rules using our simulation language, while comparing our rules to the inference rules in the sequent calculus, which is arguably the best established proof presentation system from a proof theoretic viewpoint.

### 3.1 Inferences in Natural Deduction and Sequent Calculus

Various logical proof theories have been developed for capturing the deductive inference patterns that appear in human reasoning in a precise and fully predictive manner. For example, the intuitively valid inference pattern in (12) can be captured by the natural deduction (ND) proof for classical first-order logic (FOL) in (13), with the English-to-FOL translation in (14) and the ND proof rules in (15).

(12) Every dog barks. Nero is a dog. Therefore, Nero barks.

(13) $\forall x(Dog(x) \rightarrow Bark(x))$  
    $\frac{(Dog(nero) \rightarrow Bark(nero))}{Bark(nero)}$  
    $\frac{UE}{\rightarrow E}$

(14) a. Every dog barks.  
    $\forall x(Dog(x) \rightarrow Bark(x))$

b. Nero is a dog.  
    $Dog(nero)$

c. If Nero is a dog, Nero barks.  
    $(Dog(nero) \rightarrow Bark(nero))$

(15) For all FOL formulas, $\phi, \psi$:

a. Universal Elimination (UE):  
    $\frac{\forall x\phi}{\phi[t/x]}$  
    $\frac{UE}{\rightarrow E}$

b. $\rightarrow$ Elimination $\rightarrow E$):  
    $\frac{(\phi \rightarrow \psi) \phi}{\psi}$  
    $\frac{\rightarrow E}{\rightarrow E}$

---

7 In this paper, we only discuss truth-based reasoning such as the reasoning captured by classical first-order logic (FOL), although logical systems are often motivated for characterizing other kinds of human reasoning, such as a Type Logical deductive system for natural language grammar inferences, cf. Moortgat (1997).
In (15), we have only shown the ND inference rules that we used in (13). Each proof rule in ND proceeds from one or two assumption formulas above the horizontal bar to one conclusion formula below the bar. Thus, UE in (26) means that from the assumption formula, $\forall x\phi$, we can conclude $\phi[t/x]$. $\phi$ and $\psi$ are meta-variables and since they are universally quantified in the rule specification, they can be any formulas. The conclusion, $\phi[t/x]$, means that this conclusion is like the formula $\phi$ in the assumption formula $\forall x\phi$, but with all the free occurrences of the variable $x$ inside $\phi$ being replaced by an arbitrarily chosen term $t$. Technicality aside, it basically means that if the universally quantified proposition such as $\forall x(Dog(x) \rightarrow Bark(x))$ is assumed to be true, we then can eliminate the universal quantifier while replacing all the occurrences of the variable $x$ that were bound by the universal quantifier before its elimination with a new term $t$. And this replacement term $t$ can be any term, such as $nero$ in (13). Again, this ND proof rule corresponds to the human intuition that if something is true for every dog, then it follows that that something is true for any particular dog, such as Nero, and hence you can make the rule more specific, for example, the instantiation of the universally quantified formula in (14-a) as the non-quantificational formula in (14-c).

Once we derive $(Dog(nero) \rightarrow Bark(nero))$ in the second row from the top in (13), the next step down uses $\rightarrow$E (which corresponds to Modus Ponens Ponendo (MPP) in the Hilbert-style axiomatization of the proof theory for FOL. Again, this rule is taken to correspond to the human intuition that if we assume two propositions in the forms, 1) If $P$, then $Q$ and 2) $P$, then from there, we can conclude $Q$, whatever the propositions $P$ and $Q$ are.

As we indicated, FOL and the ND proof rules as above provide a precise characterization of the deductive reasoning that humans conduct. This in turn will allows the theorist to construct a cognitively plausible computational theory of human reasoning.

Also, a logical system as above facilitates the investigation of the proof theoretic properties of deductive reasoning, such as its decidability. In fact, the ND proof system is not

---

8For a full set of ND proof rules with an accessible introduction to FOL, see van Dalen (2004).

9Natural language expressions such as John may have an additional set of connotations that are irrelevant for the sort of reasoning discussed in this paper, which may make the rule, ‘If Nero is a dog, Nero barks’, look more than just a particular instantiation of the universally quantified rule, ‘Every dog barks (or equivalently, ‘For every dog $x$, if $x$ is a dog, $x$ barks,’ which is closer to the FOL formula in (14-a)). But this paper does not discuss such subtleties with natural language semantics and pragmatics. We provide English expressions only for showing what the FOL formulae roughly mean.

10See van Dalen (2004). In fact, for both classical propositional logic and FOL, the cognitively less plausible resolution mechanism is more extensively used in their computational implementation such as one in theorem proving, mostly for some efficiency/tractability reasons. ND proofs are more extensively used for non-classical logics such as linear logic and for typed or untyped lambda calculi, see Girard (1990).
particularly good in this regard\textsuperscript{11}. Instead, consider the sequent calculus proof for the inference pattern in (12)\textasciitilde(13):

\[
\frac{\text{Dog}(\text{nero}) \vdash \text{Dog}(\text{nero}) \quad \text{Bark}(\text{nero}) \vdash \text{Bark}(\text{nero})}{\text{Dog}(\text{nero}), (\text{Dog}(\text{nero}) \rightarrow \text{Bark}(\text{nero})) \vdash \text{Bark}(\text{nero})} \rightarrow L
\]

\[
\frac{\text{Dog}(\text{nero}), \forall x (\text{Dog}(x) \rightarrow \text{Bark}(x)) \vdash \text{Bark}(\text{nero})}{\forall L}
\]

Unlike the natural deduction proof, which operates on formulas such as $\forall x (\text{Dog}(x) \rightarrow \text{Bark}(x))$ and $\text{Dog}(\text{nero})$, the sequent calculus takes as assumptions above the horizontal bar one or two \textbf{sequents} in the form of ‘Antecedent $\vdash$ Succedent,’ where both the antecedent and the succedent are sets of formulas. Also, unlike the ND proofs, which can either introduce or eliminate logical connectives such as $\forall$ and $\rightarrow$ as the inference proceeds from top to bottom, the sequent proof only introduces logical connectives from top to bottom.\textsuperscript{12} Thus, starting from the so-called ‘goal’ sequent ‘$\forall x (\text{Dog}(x) \rightarrow \text{Bark}(x)), \text{Dog}(\text{nero}) \vdash \text{Bark}(\text{nero})$’,\textsuperscript{13} the sequent proof rules as the ones in (17) below only eliminate one logical connective/operator after another from bottom to top. Since the goal sequent (that is, the statement or inference pattern that we want to prove) is generally finite, including a finite number of logical connectives and operators, and since each proof step in the sequent calculus eliminates exactly one logical operator or connective, the proof necessarily terminates in a finite number of steps upward.\textsuperscript{14} This property is called the \textbf{subformula property} of the sequent calculus and this confirms the decidability of the sequent calculus for propositional reasoning.\textsuperscript{15}

(17) shows the sequent rules used in (16).

\[
(17) \quad \text{For all formulas } \phi, \psi \text{ (and irrespective of what other formulas are hidden in } \Delta_i \text{ where } i \in \mathbb{N}.).
\]

\textsuperscript{11}See Curien and Herbelin (2000) for discussion.

\textsuperscript{12}The ND proof in (13) only uses the elimination rules, which eliminate connectives from top to bottom. But the ND proof system also has introduction rules for connectives/operators. Thus, the number of connectives may either increase or decrease either top to bottom or bottom to top.

\textsuperscript{13}Notice that this goal sequent corresponds to the entire inference pattern in (28), that is, the human inference pattern that we want to ‘prove’ formally.

\textsuperscript{14}This is not necessarily the case if we include the so-called structural rules, but there is a way of proning structural rules to maintain the decidability of the sequent calculus for classical logic.

\textsuperscript{15}The decidability of the sequent calculus for FOL requires a certain set of conditions on the variable terms.
a. Universal Left (∀L):
\[
\frac{\Delta_1, \phi \vdash \psi, \Delta_2}{\Delta_1, \forall x \phi[x/t] \vdash \psi, \Delta_2} \quad \forall L
\]

N.B. \(x\) is free for \(t\) in \(\phi\).

b. Arrow Left (→L):
\[
\frac{\Delta_1 \vdash \phi, \Delta_2, \psi \vdash \Delta_3}{\Delta_2, \Delta_1, \phi \rightarrow \psi \vdash \Delta_3} \rightarrow L
\]

In (17), ‘\(\Delta_1 \sim \Delta_3\)’ can include any formulas. Since FOL is monotonic, the rules in (17) hold no matter which formulas are included in \(\Delta_i\).\(^{16}\) Again, notice that each of the rules in (17) removes exactly one operator or connective from bottom to top, and this holds for all the rules for the logical operators and connectives in FOL, maintaining the subformula property mentioned above.

For comparison to the inference rules that we provide in our simulation language, we show the logical axioms in the sequent calculus for FOL (cf. Amon (1993), Girard (1987)).

Before showing the sequent rules specific to FOL, I first show the sequent rules for classical propositional logic, which are all inherited in the sequent calculus for FOL. The proofs here are elementary, and are a simple application of the Gentzen sequent presentation of classical logic as in Girard (1987) and Takeuti (1987).

Gentzen sequent proof representation places the sequent to prove at the bottom of the derivation. I call it the goal sequent of the proof. Then, one logical connective after another is eliminated from bottom to top, as is shown in examples below. If the proof is complete, the sequents at the top of the proof will all be in the form of (18). (20-a)~(20-d) show the rules for & and \(\rightarrow\) and ((20-e)) shows the Cut axiom. The meta-variables \(A, B\) are propositional variables \((A, B\) etc. can represent complex propositions, such as \(p \& q\) and \((p \rightarrow q) \& r\)). \(X, Y\) are structural variables for sets of propositions, such as \(\{p, q, (p \& q) \rightarrow q\}\).

\[
(18) \quad \text{Identity Axiom: } X, A, Y \vdash Z, A, W
\]

All the formula variables, such as \(A\) and all the structural variables such as \(X, Y, Z, W\) are assumed to be universally bound in these rules. That is \(A\) can be any formula and

\(^{16}\) Also, the left-to-right order between \(\Delta_i\) and \(\phi, \psi\) etc. does not matter, since both the antecedent and the succedent of the sequents in FOL are a set of formulas, and the order between the members of a set does not matter, such as \(\{a, b, c\} = \{c, b, a\}\).
X, Y, Z, W can be any set of formulas, can be a possibly empty set.

Intuitively, what the axiom in (18) validates is the argument in the form of ‘If A, then A’ (or ‘If A is true, then A is true). By convention, p, q, r, represent atomic propositional letters, A, B, C represent (propositional) formulas which can be complex formulas such as p \to q and p \land q, and X, Y, Z, W represent possibly empty sets of such formulas. For example, if the sequent notation is X, A, Y \vdash X, A, Y as in (18), it can be, ‘A \vdash A’, ‘B, A, C \vdash A’, ‘A, B \vdash D, A’, A \vdash B, C, A, etc., where A, B, C, D are propositional formulas. The commas to the left of ‘\vdash’ means ‘AND’ whereas the commas to the right of ‘\vdash’ means ‘OR’.

The left-to-right order between A, B, C, D are not important in this calculus. The possibly non-existent background formulas in X, Y, Z are included in this particular rule formulation in order to implicitly accommodate the monotonicity of classical logic, that is, when some sequent, such as ‘A \vdash A’ is valid, then the sequent stays valid no matter what formulas we add to the sequent. The validity is maintained by way of permutation between the formulas in the antecedent or in the succedent. For example, ‘A, B \vdash X if and only if B, A \vdash X’ for any A, B, X. Given this, in the rule formation in (20) below, we omit some of the structural meta-variables for readability. The reader should be aware that other than the formulas under focus, that is, the formulas meta-represented by A, B, C, D, there may appear any other formulas in the sequent without changing the validity of the rules.¹⁷

(19) shows two more axioms that do not require any assumption.

(19) a. \bot \vdash X
    b. X \vdash \top

The sequent in (19-a) means that if we assume ‘\bot’, which denotes a contradiction and therefore is always false, then we can put any formula(s) in the succedent (i.e., to the right of \vdash). This corresponds to the famous fact about CPL; if one assumes a contradiction, one can conclude anything. (19-b) means that one can always put the universally true formula \top, which is always true and hence one can freely conclude it to the right of \vdash, no matter which set of formulas are to the left of \vdash.

The axioms in (20-a)~(20-d) are the rules for the standard set of CPL connectives, \land, \lor, \to, \neg. Cut in (20-e) is an admissible rule. That is, any sequent that we can prove with Cut is provable without Cut. Thus, theoretically, Cut is not necessary for the proof

¹⁷Naturally, when the proof theoretic precision is required, the rule formulations show the structural rules such as Left/Right Weakening and Permutation in an explicit manner, but this paper does not require that level of rigidity.

17
system but Cut is useful for improving the efficiency of the proof presentation. Since CUT is an admissible rule, we ignore this rule henceforth.

\[ (20) \]

a. \[ \frac{X, A, B \vdash Y}{X, A \land B \vdash Y} \quad \frac{X \vdash A, Z, Y \vdash B, W}{X, Y \vdash A \land B, Z, W} \]

b. \[ \frac{X, A \vdash Y}{X, Z, A \lor B \vdash Y, W} \quad \frac{X \vdash A, B, Y}{X \vdash A \lor B, Y} \]

c. \[ \frac{X \vdash A, Y, B \vdash Z}{X, Y, A \rightarrow B \vdash Z} \quad \frac{X, A, Y \vdash B}{X, Y \vdash A \rightarrow B} \]

d. \[ \frac{X \vdash A, Y}{X, \neg A \vdash Y} \quad \frac{X, A \vdash Y}{X \vdash \neg A, Y} \]

e. \[ \frac{X \vdash A}{X \vdash Y} \quad \text{Cut} \]

(N.B. A in (20-e) is called the Cut formula.)

We have omitted some of the contextual structural variables (i.e. X, Y, . . .) for readability. As we discussed above, CPL is monotonic and hence, all the rules in (20) stay valid no matter more formulas we add anywhere in either the antecedent or the succedent of each sequent. Also, either in the antecedent or in the succedent of each sequent, the left-to-right order between the propositional formulas is irrelevant (i.e., \( \{\phi_1, ..., \phi_n\} = \{\phi_n, ..., \phi_1\} \)).

\( \land L \) in (20-a) is ‘pure’ in the sense that it does not introduce a new propositional variable in the inference from the top to the bottom, as opposed to the ‘impure’ inference rule from \( A \vdash X \) to \( A \& B \vdash X \) from top to bottom in the sequent calculus, which is valid in classical logic, but has incorporated structural weakening in the Antecedent. The ‘pure’ presentation as in (20) is preferable from a proof-theoretic viewpoint. Similarly, \( \lor R \) in
(20-b) is pure. In contrast, the alternative rule formation that has ‘X ⊢ A, Y’ above the horizontal bar and ‘X ⊢ A ∨ B, Y’ below the bar is NOT pure, although this alternative rule is often used as ∨R. This alternative is not pure since this alternative introduces in the sequent below the horizontal bar a new formula, B, which did not exist in the sequent above the bar. In contrast, in ∨R in (20-b) above, both A and B are present both above and below the bar (below the bar, they are connected by ‘∨’). s In (20), except for the Cut rule, the number of the connectives decreases by one along each consecutive step upwards. Because there are only a finite number of connectives in each sequent to be proved, any proof is decidable in a finite step, unless Cut is used. This is called the SUBFORMULA PROPERTY of Gentzen sequent presentation and is important for proving the decidability of the sequent calculus.

Strictly speaking, the negation rule in (20-d) is not necessary, since ¬φ in CPL is just a shorthand for the complex formula, (φ → ⊥), where ⊥ denotes ‘CONTRADICTION’ (and we already have an axiom for the contradiction formula ⊥ in (19)), but having the left and the right rules for the negative operator ¬ can make some proofs shorter.

As we stated above, all the axioms above for CPL are maintained in the sequent calculus for FOL. However, note that each pair of propositional formulas A and B are taken to be the same formulas only if all the predicate, argument terms and connectives, operators that appear in A and B are the same. For example, the formulas Red(d) and Red(f) are not the same formulas since the constant terms d and f are not the same. In contrast, since free variables in FOL may denote different objects depending on the assignments, Red(x) may have the same truth value with Red(john), when x refers to the individual denoted by the constant john. However, in this paper, we only consider the validity of the sequent in which all the free variables are implicitly universally quantified. That is, the sequent ‘X ⊢ Y’ is valid if and only if all the free variables that appear in the sets of formulas X and Y are implicitly quantified by universal quantifiers that takes the wider scope over ⊢ (for example, the sequent ‘A[x_1, x_2] ⊢ B[y]’ is valid only if for all the possible assignments to the free variables x_1, x_2, y and for all the possible interpretation models (which we discuss in the next section), whenever φ is true, ψ is also true. Because of this, we can regard the formulas Red(x) and Red(john) to be different formulas, unless some additional rules allow you to conduct some term replacement, as we see below.

Given the above, the only remaining rules we need are the rules for the quantifiers, since those are the only additional operators/connectives that FOL has on top of the operators/connectives of CPL. (21) shows those remaining rules.
(21) a.  
\[
\begin{align*}
X, A[t/x] \vdash Y & \quad \forall L \\
X, \forall x A \vdash Y & \quad \forall R
\end{align*}
\]

b.  
\[
\begin{align*}
X, A \vdash Y & \quad \exists L \\
X, \exists y A[y/x] \vdash Y & \quad \exists R
\end{align*}
\]

(*) In ∀R and ∃L, the eigenvariable x should not occur free in the conclusion sequent (that is, the sequent below the horizontal bar).

In (21), the ∀ Left rule has a different form from ∀ Left in (17) above, but they have exactly the same effect. The formulaion in (21-a) makes more sense when we evaluate the proof from the sequent below the horizontal bar to the sequent above the bar. ∀L in (21-a) means that starting with the quantified formula ∀xA below the horizontal bar placed in this structural configuration, when we move to the sequent above the horizontal bar, we can first remove the universal quantifier ∀x and then replace all the occurrences of x inside A that used to be bound by ∀x with a term t (which can be any term). To understand this rule, consider the example proof in (22).

(22)  
\[
\begin{align*}
\text{Run}(john) \vdash \text{Run}(john) & \quad \forall L \\
\forall x \text{Run}(x) \vdash \text{Run}(john) & \quad \forall L
\end{align*}
\]

The goal sequent ‘∀x Run(x) ⊢ Run(john)’ informally means, ‘If everybody runs, then it follows that John runs’, which is intuitively a valid argument, but we want to properly prove this argument is valid. Now, again according to the rule ∀L in (21-a), from bottom to top in (22), we can replace the variable x with ANY term, after removing the quantifier ∀x. In (22), we have replaced x with john, since that gets us the sequent ‘Run(john) ⊢ Run(john)’, which has the form, A ⊢ A’ and hence, is valid without any assumption, according to the axiom in (18) above. In (22), we could have replaced x with any other term, say, meg, but then the resultant sequent above the bar would have become, Run(meg) ⊢ Run(john), and there is no way of continuing the sequent calculus any further upward, since this sequent does not contain any operator or connective any more. Of course, this sequent Run(meg) ⊢ Run(john) is not valid, either (it means, ‘If Meg runs, John runs’ which can be true in the real world, but this is not logically a valid statement, since John might not run even if Meg runs. This intuition again is supported by the sequent calculus above, since it cannot prove ‘Run(john) ⊢ Run(meg)’.
∃R in (21-b) works in the same way. That is, ‘A[t/x]’ above the horizontal bar means that we can replace all the occurrences of x which used to be bound by ∃x below the bar with any term t.

In contrast, ∀R in (21-a) and ∃L in (21-b) are not ‘pure’ since we need to assign the side condition on the eigenvariable x, as we did following * be;pw (21). This could be improved (cf. Amon (1993)), but (21) is fine.

Now, here are the important facts, the sequent calculus for CPL in (18)∼(20) are sound and complete with regard to the validity in the usual Boolean algebraic interpretation (as an informal reminder, each propositional letters denote either TRUE or FALSE depending on the interpretation function, (φ ∧ ψ) is TRUE if and only if φ and ψ are both true, etc. Soundness and completeness means that all the sequents are provable by way of (18)∼(20) if and only if those sequents are valid (i.e. those sequents hold true in all the possible ‘TRUE/FALSE’ assignments to the atomic propositional letters). For a proof, see Girard (1987). Now, as for the sequent calculus for FOL, it is not complete with regard to all the well-formed formulas of FOL. However, if we focus on a certain subset of formulas, then the calculus with (18)∼(21) is sound and complete with regard to the usual first-order set based interpretation model and the interpretation rules (see van Dalen (2004), ?)), extended with the above mentioned Boolean algebra at the level of the CPL calculus. Thus, if we can axiomatize our proof theory by expressing each sequent rule above in terms of simulation with exactly the same intended interpretation, then it follows that our proof theory as well is sound and complete with regard to the same semantics as above, which are algebraically well-identified.

To state the result of this endeavor first, it does not work. Ultimately, it would be better to define our proof theory after translating our simulation language representations into Conjunctive Normal Forms (CNF) in order to run a proof theory in terms of resolution (cf. Amon (1993)). However, the proof theory in terms of resolution is not proof theoretically satisfactory since the embedding of CNFs are maximally two, as one can see in the CNF schema, ((A_1 ∨ ... ∨ A_n) ∧ (B_1 ∨ ... ∨ B_m) ∧ ..... ∧ (G_1 ∨ .... ∨ G_k) ∧ ....), where each formula A_h, B_i, G_j etc. is either an atomic predicate-argument structure Pred(t_1, ..., t_n) or its negation ¬Pred(t_1, ..., t_n) (that is, for FOL). Since higher-order structural embedding characterizes human reasoning, resolution proofs do not capture some crucial features of humans’ deductive reasoning well. Also, the above-mentioned characterization of the se-

\[^{18}\]We do not discuss the identity of this fragment of FOL, but informally, we need to add some restriction on the interpretation of free-variables, such as all the free variables are supposed to be alphabetically invariant. See ?).
sequent rules in terms of simulation works for most of the applicationally useful inference patterns, although it does not work completely. Thus, it is still meaningful to consider where this translation does not work and how we can improve this situation in the future research. We do this in the next subsection.

### 3.2 Inference in terms of simulation

First, we sketch how inferences proceed in simulation in a general manner. Then, we try to express each sequent rule that we discuss in the previous section in terms of simulation. Finally, we try and prove some of the arguments we have proved in ND or in sequent calculus in the previous section in terms of those simulation-based inference rules.

The most basic inference rule in simulation is ‘matching.’ More specifically, when all the formulas in the form of \( \text{Predicate}(t_1, ..., t_n, wib) \) (i.e., an assumption in the basis of the simulation \( wi \) where \( i \) is some natural number) is matched with \( \text{Predicate}(t_1, ..., t_n, R) \) (i.e., every component of the formulas other than \( wib \) versus \( R \) are exactly the same), then for each conclusion atom in that simulation world \( wi \), that is, for each formula in the form of \( \text{Predicate}(j_1, ..., j_m, wic) \), we can add a new formula in the form of \( \text{Predicate}(j_1, ..., j_m, R) \) in the set (again, simply replace the world argument \( wic \) with \( R \) while maintaining the rest). Consider (23) as an example. This simulation pattern is meant to capture the inference pattern in the human reasoning in (23). ‘\( \Gamma \Rightarrow \Delta \)’ means that ‘if all the contents of \( \Gamma \) are true, then we can conclude that the contents of \( \Delta \) are also true.’

\[
(23) \quad \begin{align*}
\text{a. } & \text{[If it rained, the grass is wet. It rained.]} \Rightarrow \text{[The grass is wet].} \\
\text{b. } & \{\text{Rain}(w1b), \text{Wet}(\text{grass}, w1), \ldots, \text{Rain}(R), \ldots\} \\
& \Rightarrow \\
& \{\text{Rain}(w1b), \text{Wet}(\text{grass}, w1), \ldots, \text{Rain}(R), \text{Wet}(\text{grass}, R), \ldots\}
\end{align*}
\]

In (23), the simulation world \( w1 \) is based on only one assumption, that is, \( \text{Rain}(w1b) \), corresponding to the English conditional, ‘If it rains,...’. This ‘condition’ (or assumption) is matched with a fact in the real world, that is, \( \text{Rain}(R) \), which means that it is indeed raining in the real world. Since the condition \( \text{Rain}(w1b) \) is matched with the fact in the real world \( \text{Rain}(R) \), we can add the conclusion of the simulation \( w1 \), that is, \( \text{Wet}(\text{grass}, w1) \) (‘..., the grass is wet’), by replacing \( w1 \) with \( R \), as we did to the right of ‘\( \Rightarrow \)’ in (23-b). The basis of a simulation world \( wi \) of course may contain more than one premise formula and more than one conclusion formula. Also, the ‘satisfier’ formula \( \text{Rain} \) does not need to
have \( R \) has its final argument. The essential point about this satisfier formula is that it is not an assumption. Thus, any formula that has either \( R \) or a ‘conclusion’ argument in the form of \( wic \) has its last argument can act as the satisfier formula. Accommodating these two points, the more general rule would be as in (24)

\[
(24) \quad M: \{ A_1[wib], \ldots, A_n[wib], B_1[wic], \ldots, B_m[wic], \ldots, A_1[wjc], \ldots, A_n[wjc], \ldots \} \\
\quad \Rightarrow \\
\quad \{ A_1[wib], \ldots, A_n[wib], B_1[wic], \ldots, B_m[wib], A_1[wjc], \ldots, A_n[wjc], \ldots, \\
\quad B_1[wjc], \ldots, B_m[wjc] \}
\]

N.B. The world argument ‘\( wjc \)’ can be \( R \).

All the metavariables \( A_i \) and \( B_j \) as well as all the argument term meta-variables \( wib \), \( wjc \) etc. are implicitly universally quantified, that is, those can be any formulas and any world argument terms (although the basis-conclusion distinction must be respected). The notation \( A_1[wib] \) means some atomic formula \( A_1 \) in which the argument term \( wib \) occupies a particular position. In this case, the particular position for this term is the world argument, and so this formula means that the formula \( A_1 \) is an assumption of the simulation world \( wi \). The simulation world \( wj \) may have one or more assumption/basis formulas, but we omit them inside, ‘…’ Note that all the formulas that are in the set to the left of \( \Rightarrow \) is maintained in the set to the right of \( \Rightarrow \). This means that the contents of the set increases as the inference proceeds, unless a contradiction is derived at some stage (which we discuss later), reflecting the fact that, in the actual human reasoning, the knowledge of the agent monotonically increases (again, until the agent notices some contradiction, which can lead to the revision of some of the previous beliefs, etc. as we discuss below).

As we discuss later, \( M \) in (24) corresponds to \( \rightarrow \) Elimination in ND proofs and \( \Rightarrow \) Left in the sequent calculus.

Now, the matching at (23) requires a strict match between the assumption formula, \( Rain(w1b) \), and the ‘satisfier’ formula, \( Rain(R) \). However, when the simulation implies a universal quantification, as it is the case with the simulation that implies, ‘Every dog is a mammal,’ then the assumption formula and the ‘satisfier’ formula can be also different in terms of the argument position that is occupied by the ‘newly introduced object term.’

We could define another matching rule that can realize this matching with regard to a universal quantifier simulation. However, consider how the natural deduction proof deals with the human inference in (25-a) first, which we show in (25-b).
(25) a. [Every dog is a mammal. D is a dog.] ⇒ [D is a mammal.]

b. ND proof:

\[
\forall x (\text{Dog}(x) \to \text{Mammal}(x)) \quad \forall E \quad \text{Dog}(d) \quad \frac{\text{Mammal}(d)}{\text{Dog}(d) \to \text{Mammal}(d)} \to E
\]

From top row to the second row, the ND proof in (25-b) eliminates \(\forall x\) from the universal formula, turning \(\forall x (\text{Dog}(x) \to \text{Mammal}(x))\) (= ‘Every dog is a mammal’) into a normal conditional formula \((\text{Dog}(d) \to \text{Mammal}(d))\) (= ‘If \(d\) is a dog, \(d\) is a mammal’). This \(\forall\) Elimination rule is justified since if every dog is a mammal, then it follows that for any particular individual \(x\), if \(x\) is a dog, then it follows that \(x\) is a mammal. Thus, we can simply choose the most convenient individual object term, in the above case, \(d\), and rewrite the universally quantified rule, \(\forall x (\text{Dog}(x) \to \text{Mammal}(x))\), as a more specific rule, \((\text{Dog}(d) \to \text{Mammal}(d))\). Note that the above proof has chosen \(d\) to instantiate the universally quantified rule since the other premise \(\text{Dog}(d)\) has this argument, and thus, from the second row to the bottom row, we can use \(\to\) Elimination rule (which is basically the same as Modus Ponens Ponendo) to conclude \(\text{Mammal}(d)\), as desired.

Now, so what we need in simulation is the rule that rewrites a simulation that has a universal implication to a simulation that implies a more specific conditional, such as \((\text{Dog}(d) \to \text{Mammal}(d))\) above. Consider (26).

\[
(26) \quad \{\text{New}(d, w1b), \text{Dog}(d, w1b), \text{Mammal}(d, w1c), ...\} \\
\Rightarrow \\
\{\text{New}(d, w1b), \text{Dog}(d, w1b), \text{Mammal}(d, w1c), \text{Dog}(a, wib), \text{Mammal}(a, wic), ...\}
\]

The object term \(d\) is newly introduced in the basis of the simulation world \(w1\), as the formula, \(\text{New}(d, w1b)\), indicates. Thus, the set of formulas, \(\{\text{Dog}(d, w1b), \text{New}(d, w1b), \text{Mammal}(d, w1)\}\) implies ‘Every dog is a mammal’, as we explained above. What we have added in the set to the right of \(\Rightarrow\), that is, \(\text{Dog}(a, wib), \text{Mammal}(a, wic)\), implies ‘If the object \(a\) is a dog, then \(a\) is a mammal.’ The term \(a\) that replaces \(d\) in the newly added formulas to the right of \(\Rightarrow\) can be any term. Thus, when we want a specific implication rule, such as, ‘If John is a dog, John is a mammal,’ then we simply instantiate \(d\) in (26) as \(\text{john}\). Considering these rules, (27) is our simulation rule that corresponds to \(\forall E\) in ND proofs.

(27) UI (Universal Instantiation):
\{\text{NewAt}(t_i, w_{ib}), A_1[t_i, w_{ib}], \ldots, A_n[t_i, w_{ib}], B_1[t_i, w_{ic}], \ldots, B_m[t_i, w_{ic}], \ldots\}
\Rightarrow
\Delta \cup \{A_1[t_i \rightarrow t_j, w_{ib} \rightarrow w_{jb}], A_n[t_i \rightarrow t_j, w_{ib} \rightarrow w_{jb}], B_1[t_i \rightarrow t_j, w_{ic} \rightarrow w_{jc}], \ldots, B_m[t_i \rightarrow t_j, w_{ic} \rightarrow w_{jc}], \ldots\}

where

\begin{itemize}
  \item \(w_{ic} \neq R, w_{jc} \neq R\)
  \item \(\Delta = \{\text{NewAt}(t_i, w_{ib}), A_1[t_i, w_{ib}], \ldots, A_n[t_i, w_{ib}], B_1[w_{ic}], \ldots, B_m[w_{ic}], A_1[t_j, w_{jc}], \ldots, A_n[t_j, w_{jc}]\}\)
\end{itemize}

The notation \(A_1[t_i \rightarrow t_j, w_{ib} \rightarrow w_{jb}]\) represents the formula \(A_1\) in which the term \(t_i\) is replaced by the term \(t_j\) and the term \(w_{ib}\) is replaced by the term \(w_{jb}\). If there is more than one occurrence of \(t_i\) inside \(A_1\), then all of its occurrences are replaced by \(t_j\). Each formula has exactly one world argument so there is exactly one \(w_{ib}\) in \(A_1\), which is replaced by \(w_{jb}\) in the above formula.

As we indicated above, UI in (27) corresponds to \(\forall\) Elimination in ND proofs, which in turn corresponds to \(\forall\) Left in the sequent calculus.

Using M in (24) and UI in (27), we prove the argument in (25-a), corresponding to the ND proof in (25-b), and essentially the same as the sequent proof in (16) above, which we repeat in (28-b), modifying it to correspond to the argument in (25-a).

(28)
\begin{itemize}
  \item a. [D is a dog, Every dog is a mammal.] \Rightarrow [D is a mammal.]
  
  \item b.
\[
\frac{
\text{Dog}(d) \vdash \text{Dog}(d) \quad \text{Mammal}(d) \vdash \text{Mammal}(d)
}{
\text{Dog}(d), (\text{Dog}(d) \rightarrow \text{Mammal}(d)) \vdash \text{Mammal}(d) \quad \Rightarrow L
}
\]
\[
\frac{
\text{Dog}(d), \forall x(\text{Dog}(x) \rightarrow \text{Mammal}(x)) \vdash \text{Mammal}(\text{nero})
}{
\forall L
}
\]
  
  \item c. \{\text{Dog}(d, R), \text{NewAt}(t, w_{1b}), \text{Dog}(t, w_{1b}), \text{Mammal}(w_{1c})\}
\Rightarrow \text{UI}
\{\text{Dog}(d, R), \text{Dog}(d, w_{2b}), \text{Mammal}(d, w_{2c}), \text{NewAt}(t, w_{1b}), \text{Dog}(t, w_{1b}), \text{Mammal}(w_{1c})\}
\Rightarrow \text{M}
\{\text{Dog}(d, R), \text{Mammal}(d, R), \text{Dog}(d, w_{2b}), \text{Mammal}(d, w_{2c}), \text{NewAt}(t, w_{1b}), \text{Dog}(t, w_{1b}), \text{Mammal}(w_{1c})\}
\]
\end{itemize}

Again, the sequent calculus in (28-b) puts the entire argument in (28) as the goal sequent at the bottom row in the proof. It then tries and remove one connective after another.
upward. In (28-b), \( \forall x \) is removed from the bottom row to the mid row, and then from the mid row to the top row, \( \rightarrow \) is removed. Since the number of connectives and operators that appear in the goal sequent is finite, and neither new connectives/operators nor new formulas are introduced from bottom row to top row (again, this is called the subformula property), the sequent calculus is decidable.

Now, unlike the sequent calculus, our simulation proof in (28-b) cannot easily start from the bottom set. This is because in this inference, the bottom set may contain something other than the argument statement that we want to prove. In the bottom set in (28-c), the two formulas, \( \text{Dog}(d, w2b), \text{Mammal}(d, w2c) \), which together implies, \( (\text{Dog}(d) \rightarrow \text{Mammal}(d)) \), does not correspond to a proposition in (28-a). Thus, like a natural deduction proof, our simulation proof will proceed from top to bottom. Just like a ND proof, the top set contains the formulas that represent the assumptions in (28-a), that is the assumptions, ‘D is a dog’ and ‘Every dog is a mammal.’ The simulation proof then tries to derive the conclusion of that statement, that is, ‘D is a mammal.’ That is achieved in the bottom set. That is, the bottom set (28-c) contains the formula \( \text{Mammal}(d, R) \), which means ‘D is a mammal’ and so the proof is complete there.

Now, although the above simulation proof does not have the elegant subformula property that the sequent calculus has, between UI and M, it is decidable, since the problem from the top set is essentially a problem of finding an instantiation of the ‘universal’ term, \( t \) with another argument term that appears in the same set (in this case, the solution is achieved by instantiating \( t \) as \( d \)).

Note that using the older language in 2.1, the depth of matching is always one.

(29) Matching in the older form (cf. M in (24)).

In Reality: \( \{A_1, \ldots, A_n\} \)

\( S_i \), based on: \( \{A_1, \ldots, A_n\} / \text{Con}: \{B_1, \ldots, B_m\} \)

In Reality: \( \{A_1, \ldots, A_n, B_1, \ldots, B_m\} \)

Of course, each assumption \( A_i \) can refer back to another simulation with \( \text{Hold}(S_j) \), and this other simulation may represent a proposition of a higher-depth, such as \( ((P \rightarrow Q) \lor R) \), etc. in terms of the inferences to be captured, the depth of inferences can be as deep as classic logic, but once a particular simulation-term \( S_i \) is assigned to a simulation that can express potentially a deep propositional embedding, then each matching as in (29) is done by matching each formula in the form of \( A_i \), and thus, the embedding depth of matching in that sense is still always one.
Below, we try to match each sequent rule in (18) with a corresponding inference pattern in simulation, repeating each rule in a. for comparison. Again, these inferences are supposed to be valid irrespective of what X, A, Y, etc. are.

(30)  

b. Identity simulation: {..., A(wib), A(wic), ...}

The context structure variables X, Y, Z, W in the sequent rule is represented as ‘...’ in the simulation rule. The simulation world wi shows that whatever assumption one puts in the basis can also be concluded as a conclusion atom. Which is obvious since if one assume that something is true as a hypothesis, that something is of course true by hypothesis, as we can see with ‘If A, then A,’ which is obviously a tautology without any further assumption.

(31)  

a. ⊥ ⊢ X
b. X ⊢ ⊤

(32) (31) in Simulation.

a. Bot rule, F: {⊥(wib), ...}
b. Top rule, T: {⊤(wic), ...}

The contradiction simulation F in (32-a) means that if one assumes a contradiction ⊥ in the basis of a simulation, that simulation can conclude everything, as it is the case with CPL reasoning, since contradiction formula is always false, and each ‘hard constraint’ simulation implies the material implication relation holds between the basis and the conclusion, as we explained in section 2. Similarly, the top rule in (31-b) means that one can include the top formula in the conclusion of any simulation, irrespective of what the basis contains (or even when the basis is totally empty). Both the simulations in (31) are valid in our simulations with regard to the interpretation that is essentially the same as that for CPL, and thus, we can maintain the validity of the two sequent rules in (31) by way of simulations with the same intended interpretations.

(33)  

a.  

\[
\frac{X, A, B \vdash Y}{X, A \land B \vdash Y} \land L \\
\frac{X \vdash A, Z \quad Y \vdash B, W}{X, Y \vdash A \land B, Z, W} \land R
\]
b. \[
\begin{align*}
X, A \vdash Y & \quad Z, B \vdash W \\
\frac{}{X, Z, A \lor B \vdash Y, W} \qquad \lor L \\
X \vdash A, B, Y & \quad \lor R
\end{align*}
\]

c. \[
\begin{align*}
X \vdash A & \quad Y, B \vdash Z \\
\frac{}{X, Y, A \rightarrow B \vdash Z} \rightarrow L \\
X, A, Y \vdash B & \quad X, Y \vdash A \rightarrow B \rightarrow R
\end{align*}
\]

d. \[
\begin{align*}
X \vdash A, Y & \\
\frac{}{X, \neg A \vdash Y} \neg L \\
X \vdash A & \quad X, \neg A \vdash Y \rightarrow R
\end{align*}
\]

e. \[
\frac{X \vdash A}{X \vdash Y} \text{ Cut}
\]
⇒
{..., A(w2b), ⊥(w2c), B(w3b), ⊥(w3c), Hold(w2, w4b), Hold(w3, w4b), ⊥(w4b), ...}

f. → L, basically, M in (24).
g. → R,
 {..., A(w1b), C(w1c), ...} 
⇒ 
{..., A(w1b), C(w1c), ..., Emptyset(w2), Hold(w1, w2c), ...}
h. ¬ L,
{..., C(w1c), ...} 
⇒ 
{..., C(w1c), ..., C(w1, w2b), ⊥(w2c), Hold(w2, w3b), Fail(w1, w3c), ...}
i. ¬ R,
{..., C(w1b), ⊥(w1c), ...} 
⇒ 
{..., C(w1b), ⊥(w1c), Hold(w1, w2b), ⊥(w2c), EmptyBasis(w3b), C[w1b :→ w3c], ...}

The syntactic sugar ‘EmptyBasis(wjc)’ specifies that the basis of the simulation world wj is empty. We of course can check whether the basis of each simulation world wn is empty or not by checking if there is a formula that has wnb as its world argument, so this syntactic sugar is only for the convenience of rule presentation. Similarly, the syntactic sugar ∆(wic) is a shorthand for C1(wic), ..., Cn(wic), which represents one or more propositions C1, ..., Cn in the conclusion of the simulation world wi. What (34-a) means is that the conclusions of the simulation world wi is the same in the sets to the left and to the right of ⇒. Also, we did not show the conclusion(s) of the simulation world wi in (34-a) since the conclusion can be anything (although in our simulation analysis, the conclusion cannot be empty). The simulation pattern for ∧ L is valid because in our analysis, the simulation based on the assumptions A and B has the same intended interpretation as the simulation based on the assumption that another simulation wj holds where this simulation wj has the propositions A and B in the conclusion (note that the simulation wj implies (A ∧ B) or ‘A and B’ in classical logic). In other words, whatever one can conclude in a simulation based on ‘A and B’ can be concluded in a simulation based on the assumption that the simulation for ‘A ∧ B’ holds, and vice versa, given how our simulations are intended to be interpreted. Because the set to the left of ⇒ and the set to the right of ⇒ in (34-a) have exactly the same implication in simulations, together with
(30-b), it is easy to see that simulations can express $\land L$, as we did in (34-b), capturing the same intended Boolean algebra interpretation.

However, since the implication ‘$\leftrightarrow$’ in (34-a) goes both ways, in (34-b), we could have monotonically increased the set right to left, rather than left to right, expressing the equivalent of switching the top row and the bottom row in the sequent rule $\land L$ in (33-a). If we take this property literally, it would violate the equivalent of the subformula property that we have already discussed, which is essential for proving the decidability of the sequent calculus. However, this sort of potentially infinite inference possibility is actually implicitly present in Classical Propositional Logic (see Uchida (2007)), which the sequent calculus has cleverly suppressed. Thus, although our first attempt has failed to capture the subformula property of the sequent calculus for CPL, it is a matter of considering a different set of axiomatization that works better in this regard.19

The sequent rule $\land R$ is shown in (34-c). Note that this simulation pattern is ‘pure’ in the sense we explained above, that is, from the right set to the left set across ‘$\Rightarrow$’ no formulas are added. We have the reader to check the translation of the other sequent rules into our simulation inference patterns, while giving some minimal comments. We have named each rule in the same way between the sequent calculus and the simulation for facilitating the comparison.

In (34-d), $\Delta_1(w1c)$ represents the set of conclusion formulas for the simulation $w1$, that is, $\{C_1(w1c), ..., C_n(w1c)\}$, and similarly, $\Delta_2(w2c) = \{C_1'(w2c), ..., C_m'(w2c)\}$. To the right of $\Rightarrow$, $\Delta_1[w1c \Rightarrow w5c] \cup \Delta_2[w2c \Rightarrow w5c]$ represent the union of the above two conclusion sets, while all the occurrences of the world argument term $w1c$ and the world argument term $w2c$ are replaced by $w5c$. Basically, this has the effect of repeating all the conclusions in $w1$ and $w2$ in the conclusion of the newly added simulation $w4$. The reason why our $\lor$ Left is so long and complicated is that our treatment of the formula $(A \lor B)$ is done only by way of $\neg(\neg A \land \neg B)$, as we have discussed in section 2. Actually, the ‘impure’ rule for $\lor L$ can be represented much simpler. $\lor$Right can be represented only in this impure manner, as (34-e) shows. $\rightarrow$ R in (34-g) is not a faithful translation of the sequent rule, $\rightarrow$ Right, although it has a similar effect. So we would need to check whether the entire intended semantics stays the same between the sequent calculus and the simulation in terms of this translation. The simulation treatment of $\neg L$ in (34-h) is

---

19 Again, obviously, there is a way of using a resolution-like inference, translating all the formulas into a maximally two-depth embedding structure, removing all the formulas that refer back to other simulations, e.g., $Hold(S_i)$, etc. (Cf. Cassimatis, Murugesan, and Bignoli (2009)). This is possible, and this is practically how we run inferences at the moment, but as we discussed above, there are several empirical and proof theoretical demerits in this analysis.
not pure. It is also not a nice formulation; what it does left-to-right is the following: the set to the left of ⇒ means that the agent concludes C (since that is the conclusion of the simulation w1). However, to the right of ⇒, the agent adds another simulation, w2, with the new formulas, C(w1, w2b), ⊥(w2c). As we explained, this simulation implies ¬C. Since this new simulation w2 contradicts the implication of the older simulation w1, the agent can clobber this entire knowledge-base and do it all over again. However, the agent could alternatively withdraw one of the simulations w1, or w2, since maintaining both means that no possible interpretation model can be constructed. In (34-h), we have shown an example in which the agent chooses to maintain w2, putting the assumption, Hold(w2, w3b) in the basis of the simulation w3. Since this last simulation w3 confirms the implication of w2, that is, ¬C, then the agent needs to conclude the withdrawal of the simulation w1, that is, Fail(w1, w3c), in the conclusion of w3. This confirms the agent’s current belief, ¬C. The agent could do the opposite, that is, withdrawing w2, leading to reconfirming the truth of C. ¬R in (34-i) does this job. Again, the simulation rule in (34-h) does not strictly correspond to ¬L in (33-d). ¬R in (34-i) does not strictly correspond to ¬R in (33-d), either, although given the general monotonicity of our simulations, they can derive the same effect that the sequent rules for ¬ achieve. We leave a better formation rules for the negation for another paper.

From the translation attempt at (33) from sequent calculus to the inference rules in simulation, we can say that translating ‘Right’ (R) rules other than the one for ∧ (i.e., ‘AND’) is not generally straightforward, which is not of course surprising, since the proposed simulation language does not literally contain any connectives, other than ‘COMMAS’ (which mean AND). However, we have also found out that the basic effect of each rule in the sequent calculus can be roughly reflected in the simulation reasoning, which is promising. As we have explained above, the sequent calculus is sound and complete with regard to the classic Boolean algebraic interpretation and also its extension to FOL with the alphabetical invariance constraint on the variable interpretation is sound and complete with regard to the standard first-order set-theoretic interpretation models, and thus, we have shown that we can almost treat these semantic models as our semantic models as well.

However, other than solving the above mentioned less than faithful translation problem from the sequent calculus into simulations, there is one remaining problem to solve, that is, our language needs to mark some terms as ‘New’ (or equivalently, some terms do not appear anywhere else in the set, {...}). Whichever characterization we may adopt, either the newness or the uniqueness of the term is not necessarily maintained in the current
simulation framework, which can potentially make the intended semantic models more complex, since it would need to have the power of keeping track of which objects are newly introduced (or unique) at each stage of monotonic model development. However, we can solve this problem by using terms that will never lose their uniqueness no matter how many object (or terms denoting objects) are introduced later. Basically, this amounts to introducing the denotations that corresponding to ‘the terms for quantified noun phrases’ in our interpretation models. See Uchida and Cassimatis (2010) for such a model structure that contains the individual objects that quantified noun phrases such as every boy and some girls can refer to.

4 Conclusion

This paper has proposed a revised simulation language that still maintains the characteristics of our simulation language. This language achieves the goal of removing the abstract elements from our language representation better than our older language. We have then tried to translate the rules of the sequent calculus for classical CPL/FOL in terms of our language, since the success of this attempt would imply that our simulation system would be sound and complete to the well-investigated semantics of these classical languages. Our first try was reasonably successful but not quite, which means that there is some room for improvement in our inference rule formations in simulation. Also, the intended semantics for our simulation language might need to include some new kind of individuals, in order to incorporate our treatment of quantifiers by way of ‘newly introduced individual terms.’

References


