From Casual Deduction to Spatial Relations: Bottom-up and Top-down Reasoning Unified.

Enkhbold Nyamsuren (e.nyamsuren@rug.nl)
Niels A. Taatgen (n.a.taatgen@rug.nl)
Department of Artificial Intelligence, University of Groningen,
Nijenborgh 9, 9747 AG Groningen, Netherlands

Abstract

This paper introduces a framework of the human deductive reasoning and its ACT-R based implementation called Human Reasoning Module. HRM provides a unified view of rule- and mental model-based strategies of deductive reasoning. The paper explores the role of bottom-up visual processes and implicit knowledge in mental model-based strategy. The validity of HRM is tested with cognitive models of two tasks involving casual deduction and reasoning on spatial relations. Via exploration through these models, it is shown how HRM combined with ACT-R's declarative memory give rise to a pragmatic reasoning.

Keywords: deductive reasoning; rules; mental models; bottom-up reasoning; ACT-R.

Introduction

In this paper, we introduce a framework of the human deductive reasoning that incorporates semantics at both top-down and visual bottom-up layers. **H**uman **R**easoning **M**odule, or HRM, is an implementation of this framework developed as a part of ACT-R cognitive architecture (Anderson, 2007).

Studies suggest two types of deductive reasoning strategies: deduction rules and mental models. The theory based on deduction rules argues for a mental logic that applies a set of deduction rules to logical forms abstracted from stimuli (Rips, 1983). The alternative theory based on mental models dictates that stimuli are abstracted into some form of mental diagram where configuration information reflects relationship between entities (Johnson-Laird, 1983). Latter studies indicated that two theories are not mutually exclusive (Roberts, 1993). It was suggested that a strategy most suitable for the task is used at the time.

Despite extensive research, little is known about the form of cognitive processes that provide meta-control over these two strategies. Furthermore, there is a distinct lack of understanding how one reasoning strategy is chosen over another depending on tasks. HRM introduces a simple, but effective cognitive construct referred to as a *reasoning pipeline* that addresses these issues. Next, despite ever-increasing evidence supporting mental models, there is still a lack of understanding of inner workings of mental models. Specifically, question remains on how semantics is extracted from mental models. We propose a visual bottom-up reasoning (not to be confused with inductive reasoning) mechanism as a primary means of extracting semantics. We define bottom-up reasoning as an ability to extract explicit knowledge from implicit knowledge via bottom-up cognitive processes. Knowledge is implicit if it is available, but we are not consciously aware of it.

This paper concentrates on visual bottom-up reasoning. Studies suggest that bottom-up visual processes may be much smarter than previously thought (Rensink, 2007). Bottom-up visual processes may be able to process information at a semantic level subconsciously and even pre-attentively. It is assumed that the mental model is a form of a visual memory that is an extension to a working memory. Such visual memory can contain either objects that were encoded from real world or abstractions of our imagery capability (Wintermute, 2012). Information about object's inherent properties, such as position, shape and color, may also be present in visual memory in a form of implicit knowledge. Such information can be readily extracted with bottom-up visual processes and converted into explicit knowledge on which top-down reasoning can operate.

Architecture of HRM

Knowledge representation in declarative memory

Concepts and triples The atomic unit of knowledge in HRM is a *concept*. Any unit of knowledge that has distinct semantic meaning can be a concept. There are two types of concepts in HRM: property instance and class instance. Property instance is any concept that is used to relate semantically two other concepts. As such, the knowledge organization inside HRM revolves around a predicate construct referred to as a *triple*: *(property subject object)*. Inside a triple, *property* establishes a semantic connection between *subject* and *object*. The following is an example of a triple: (*r-left-of fork plate*). In HRM, *r-left-of* is a property instance that is used to represent a spatial relation between two class concepts. In example above, the meaning of the triple is equivalent to "*a fork is in left side of a plate*".

A property instance can also be used as triple's subject or object. For example, HRM has two different property instances, *r-left-of* and *r-dir-left-of*, for expressing a similar spatial relation between two class instances. *r-dir-left-of* expresses semantically more restrictive spatial relation implying that subject is to the left of an object, and both subject and object are aligned vertically. Therefore, triple *(r-dir-left-of fork plate)* entails triple *(r-left-of fork plate)*. One way to express such one-way relation is to have another

triple *(entails r-dir-left-of r-left-of)*. Here, property instance *entails* semantically connects two other property instances.

Most of the studies of human mental logic advocate for some form of predicate construct as a way of knowledge organization. We have chosen the triple form because it closely resembles a linguistic typology consisting of subject, verb and object. It is the most common sentence structure found across different languages. Such commonality strongly indicates that underlying knowledge from which a sentence is constructed may also be organized in the same form consisting of subject, object and verb (Crystal, 1997).

HRM provides functionality for a modeler to create custom property and class instances. It is also possible to create custom triples. However, custom triples have to be bound to one of the triple types built into HRM.

Statements In HRM, *statement* is a type of triple that represents a factual knowledge. It is a statement of a fact that is true or was true. Triples from preceding subsection are all valid *statements*. HRM provides several ways to create a *statement*. Firstly, modeler can explicitly define custom *statements*, as model's background knowledge. Secondly, model itself can create *statements* in real-time via production rule calls to special *reasoner* buffer. This option simulates an ability to obtain a new explicit knowledge through external input, such as stimuli from an outside world. Finally, a model can generate a new statement by inferring it from existing statements using top-down reasoning, or by deriving it from an implicit connection between concepts using bottom-up reasoning.

Implicit and explicit knowledge HRM makes a distinction between explicit and implicit knowledge. *Statements* are explicit knowledge, form of a knowledge that is known consciously. Implicit knowledge is a knowledge that is represented by slot values of concept chunks. Such knowledge is implicit because it is assumed that ACT-R is consciously not aware of its presence, but subconsciously can extract it to form explicit *statements*.

Inference rules In HRM, rules describe how a new *statement* can be inferred from existing *statements*. Rules use special triples called rule-statements. Semantically, a rule-statement is not a fact, but either a condition or an implication of a possibility. Any rule consists of left- and right-hand sides. A left-hand side must have one or more rule-statements (antecedent), and the right hand-side should have exactly one rule-statement (consequent). In order for a consequent to be true, all antecedent rule-statements should also be true. For example, the rule below states "*if the fork is on the left of the plate then the plate is on the right of the fork"*:

(r-left-of fork plate) ==> (r-right-of plate fork)

Unlike ordinary *statements*, rule-statements can use variables as one of the entities in the triple. The previous example rule can be rewritten as:

(r-left-of "@item" plate) ==> (r-right-of plate "@item")

Above rule states "*if any item is on the left of the plate then plate is on the right of that item"*. In this rule, *"@item"* is a variable, not a *concept*. It can be replaced by any valid concept that is factually on the left side of the plate. Variables provide a possibility to generalize rules beyond a scope of a particular concept or even an entire model. It also introduces a possibility to reuse the same rules across different ACT-R models, at least partially, addressing one of the major reusability challenges in ACT-R.

Assertion Assertion is another type of triple used by HRM. Assertion represents a query questioning HRM whether a triple is true. For example, the assertion *(r-right-of plate fork)* represents the query: "*Is plate on the right side of the fork?*" Similar to rule-statements, assertions can have variables. The assertion *(r-right-of plate "@item")* asks HRM to find any class instance that is on the right side of the *plate*. In ACT-R, HRM can be queried with an assertion via *reasoner* buffer. Upon receiving an assertion, HRM starts a reasoning process called a *reasoning pipeline*. The task of reasoning pipeline is to check if assertion can be proven to be true or to find/prove any *statement* that matches the assertion if assertion contains variables. If assertion is true then it is converted into a *statement* and placed inside *reasoner* buffer.

Reasoning pipeline

The true power of HRM comes from its ability to generate a new knowledge from existing one, either explicit or implicit. For this purpose, HRM uses a *reasoning pipeline*. As it was discussed earlier, new knowledge can be generated from existing knowledge using one of several different strategies. The reasoning pipeline establishes priority among those strategies and organizes them into series of consecutive steps. The highest priority strategy receives an assertion first and tries to prove it. If it fails then the assertion is passed to the next highest priority strategy. In ACT-R architecture, reasoning pipeline is implemented as a series of automated calls to production rules built into HRM. HRM triggers calls to these production rules as soon as it receives an assertion request inside *reasoner* buffer. This set of production rules recursively calls itself until either the assertion is proven or it is decided that the assertion cannot be proven.

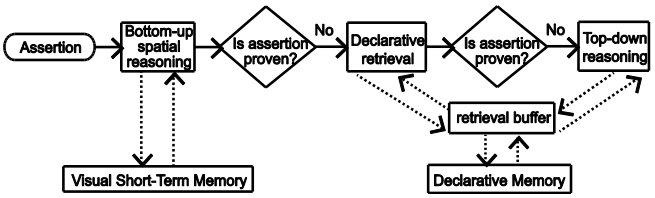

Figure 1: A simplified workflow of a HRM reasoning pipeline in ACT-R.

In its current implementation, HRM supports three different strategies of reasoning: bottom-up reasoning, declarative retrieval and top-down reasoning. Figure 1

shows the prioritization of those strategies. The bottom-up reasoning is the most preferred one requiring the least amount of cognitive effort. The bottom-up reasoning is followed by the declarative retrieval and the top-down reasoning in a decreasing order of priority.

Bottom-up reasoning The current implementation of HRM's visual bottom-up reasoning supports only spatial reasoning. As with other forms of reasoning, spatial reasoning requires source of knowledge based on which it can derive a new knowledge. In HRM, such knowledge source is a visual short-term memory (VSTM). VSTM was introduced by the newer version of PAAV (Nyamsuren & Taatgen, 2013), an extension to ACT-R's default vision module. VSTM is a high resolution, but low capacity visual memory. Every visual object encoded from an external world is temporarily stored inside VSTM until it decays out or deleted due to capacity limit. Unlike declarative memory, VSTM is considered as a visual analog of a working memory. Hence, objects inside VSTM can be accessed by HRM with no cognitive cost, and explicit knowledge can be derived with little effort.

HRM can take advantage of VSTM whenever it receives an assertion about spatial relation between two concepts such as *(r-right-of plate fork)*. VSTM contains detailed information about each visual object currently in its store, including object's original position in real world. In ACT-R, those are two-dimensional spatial coordinates. HRM can use such implicit knowledge to derive quickly explicit *statements* about spatial relations between concepts inside VSTM. If one of those derived statements supports the assertion then assertion is proven.

Declarative retrieval If bottom-up reasoning fails then HRM tries to retrieve from declarative memory any *statement* that can directly confirm the assertion. In ACT-R, a declarative retrieval can be a time-costly process. Furthermore, there is a chance that retrieval will fail even if matching *statement* exists. Those are the reasons why bottom-up reasoning takes priority over declarative retrieval as a more reliable and faster process.

Top-down reasoning It is invoked only if declarative retrieval fails. It is a rule-based reasoning where a chain of inference rules is used to prove an assertion.

Current implementation of HRM supports fully functional backward-chaining algorithm implemented as a set of ACT-R production rules. The first production retrieves from declarative memory any consequent rule-statement that matches the assertion. If a retrieval of rule's consequent is successful then the next production retrieves the first antecedent rule-statement of the same rule. The retrieved antecedent rule-statement is converted into an assertion and fed back to HRM. This starts a new recursive call with a new reasoning pipeline. If recursive call was able to prove that current antecedent rule-statement is true then the next antecedent rule-statement is retrieved, converted into assertion and fed back to HRM. This process continues until all antecedent rule-statements are proven. In such case, consequent rule-statement is also true, and, hence, the original assertion is true as well. If any of the antecedent rule-statements cannot be proven then HRM stops the reasoning process and sets *reasoner* buffer to an error state.

The top-down reasoning is a set of production calls coupled with frequent declarative retrievals. Not only it is a hugely time-consuming process, but also it is very costly in terms of cognitive resources. Since ACT-R allows only one production call at the time, it creates a bottleneck for other task-specific productions. Furthermore, declarative memory is locked through entirety of the time HRM uses it to prove an assertion. Hence, other cognitive processes cannot access declarative memory. The overall high cost puts top-down reasoning in the lowest priority position.

Validation Models

Model of Casual Deduction Task

Cummins, Lubart, Alksnis and Rist (1991) and Cummins (1995) extensively studied this task. Subjects are provided with a sentence describing a cause/effect in a form of "*If <cause>, then <effect>*". The sentence is followed by four different forms of arguments: Modus Ponens (MP), Affirming the Consequent (AC), Modus Tollens (MT) and Denying the Antecedent (DA). Each argument consists of a fact and implication. Subjects are asked to evaluate how likely it is that implication is true given cause/effect sentence and argument's fact. Here is an original example from Cummins et al. (1991) of a cause/effect sentence: "*If the brake was depressed, then the car slowed down.*". The four arguments with respect to this sentence are: "*The brake was depressed. Therefore the car slowed down.*" for MP; "*The car slowed down. Therefore the brake was pressed.*" for AC; "*The car did not slow down. Therefore, the brake was not depressed.*" for MT; and "*The brake was not depressed. Therefore, the car did not slow down.*" for DA.

The study revealed that acceptance of arguments is influenced significantly by subject's previous experience. The casual deduction was sensitive to two factors: alternative causes and disabling conditions (Cummins et al., 1991). Alternative cause is a cause that is different from a one given in a sentence but still can result in the same effect. Disabling condition is a condition that prevents the effect from occurring despite the presence of a cause. Figure 2 shows the acceptance ratings of four conditions gathered from two separate studies. Firstly, it is not surprising that acceptance rating varies a lot between two studies. The nature of the task is extremely subjective and participants' previous experiences vary a lot. Secondly, there is a robust effect of disabling conditions on acceptance of MP and MT arguments. When there are many possible disabling conditions, subjects are less likely to accept truthfulness of these two types of arguments. Thirdly, there is a persistent effect of alternative causes on acceptance of DA and AC arguments. When there are many possible alternative causes

of the effect, subjects are less likely to accept DA and AC arguments.

Using ACT-R model that uses HRM's knowledge structure, we explore the nature of effects invoked by alternative causes and disabling conditions on our ability of casual deduction.

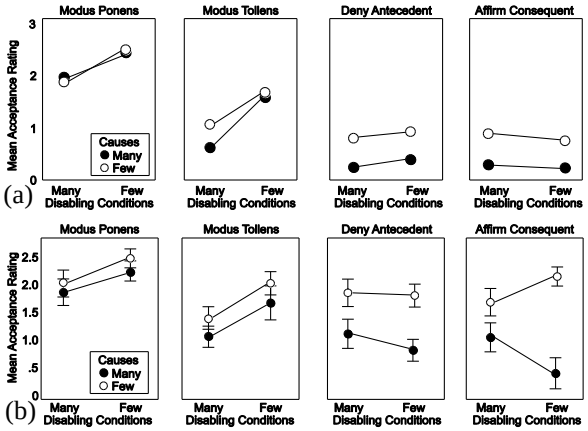

Figure 2: Mean acceptance ratings of four argument forms in casual deduction experiments conducted in (a) Cummins et al. (1991) and (b) Cummins (1995).

Model's knowledge structure In this experiment, the model used the same 16 cause/effect sentences described in Cummins (1995). The model stored both affirmative and negatives versions of all 16 sentences in its declarative memory in form of rules. For example, the previously mentioned example cause/effect sentence was converted to following two rules:

Rules 1 and 2:
(have-state brake pressed) ==> (decrease car speed)
(NOT-decrease car speed) ==> (NOT-have-state brake pressed)

Inside declarative memory, the model also had alternative causes and disabling conditions for each sentence. They were also stored in form of rules. Here is an example of affirmative and negative rules for an alternative cause:

Rules 3 and 4:
(have-state car go-uphill) ==> (decrease car speed)
(NOT-decrease car speed t) ==> (NOT-have-state car go-uphill)

An affirmative version of the same disabling condition can be written as two following rule forms:

Rule 5:	*Rule 6:*
(have-state brake pressed)	*(have-state brake pressed)*
(have-state brake broken)	*(NOT-decrease car speed)*
==>	*==>*
(NOT-decrease car speed)	*(have-state brake broken)*

Both forms were stored in declarative memory. Finally, an example of a negative version of a disabling condition would be as following:

Rule 7:
(have-state brake pressed)
(NOT-have-state brake broken)
==>
(decrease car speed)

Sentences were divided into four groups. In Many/Many group, a sentence had three disabling conditions and three alternative causes. In Many/Few group, there were three disabling conditions and one alternative cause. Similarly, the other two groups were Few/Many and Few/Few.

Model's reasoning strategy With each sentence, the model had to do four trials, one for each argument form. Model's general strategy was simple: given an argument, retrieve any matching rule from declarative memory and verify if the rule supports the argument. Depending on the argument form, the model used different forms of reasoning. For MP arguments, the model did forward reasoning with fact. It retrieved any rule that had antecedent rule-statement matching the fact and checked if retrieved rule's consequent matched the implication. If a match was found, then the argument was accepted. For AC arguments, the model did backward reasoning with fact: it retrieved any rule that had consequent matching the fact and checked if any of the antecedent rule-statements matched the implication. If match was found then argument was accepted. In a similar manner, the model did forward reasoning with fact for MT arguments and forward reasoning with implication for DA arguments.

Results The model repeated the same experiment 50 times, accounting to total of 3200 trials. Figure 3 shows proportions of trials where arguments were accepted. The proportions were calculated separately for each combination of four argument forms and sentence groups. The model shows the same behavior as human subjects. The model is more likely to accept MP and MT arguments for cause/effect rules that have few disabling conditions. Next, the model is more likely to accept DA and AC arguments for cause/effect rules that have few alternative causes.

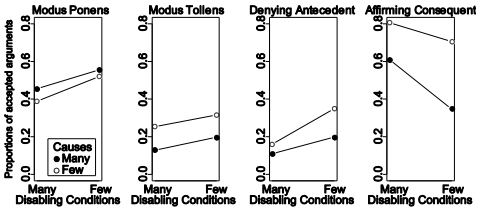

Figure 3: Proportions of arguments accepted by the model in four forms of arguments.

The effects are explained by a mutual interference among rules during the step when the model tries to retrieve a proper rule that can support an argument. For example, let us assume that the model received following MP argument:

Fact: (have-state brake pressed)
Implication: (decrease car speed)

In this scenario, the model will use the fact *(have-state brake pressed)* to retrieve any rule with matching antecedent rule-statement. These include not only the original cause/effect rule 1, but also affirmative and negative disabling condition rules 5, 6 and 7. In presence of several matching chunks during a declarative retrieval, ACT-R

randomly picks one. The rules 5 and 6 have consequents that are different from the argument's implication. Therefore, if either rule 5 or 6 is retrieved then the model will not accept the argument's implication. It is quite easy to see that as the number of disabling conditions increases, the model will be less likely to retrieve a rule that supports the argument and, hence, more likely to reject it. This rule interference mechanism is also responsible for the effects observed in other three argument forms.

Model of Spatial Relations Task

This task is used to study people's fundamental ability to derive a spatial relation from a set of premises. Three problems below are examples of such task. In each problem, subject is given four premises and then queried about spatial relation between two items that were not explicitly connected in any of the premises.

The studies showed that people prefer to use strategy of mental states rather than formal representations (Byrne & Johnson-Laird, 1989). In such strategy, people build mental states or imagery using abstract objects representing items in the premises. Such mental state is built iteratively as premises are processed one by one (Carreiras & Santamaria, 1997). With such mental states, the spatial relation between two query items can be derived directly. Examples of mental states are shown below. Problem 1 results in a one metal state. Problems 2 and 3 result in two possible mental states. Furthermore, the same studies have shown that one-state problems are easier than two-state problems.

Problem 1:
1. A is on the right of B
2. C is on the left of B
3. D is below C
4. E is below B
What is the relation between D and E?

Possible mental state:
C B A
D E

Problem 2:
1. B is on the right of A
2. C is on the left of B
3. D is below C
4. E is below B
What is the relation between D and E?

Possible mental states:
C A B A C B
D E D E

Problem 3:
1. B is on the right of A
2. C is on the left of B
3. D is below C
4. E is below A
What is the relation between D and E?

Possible mental states:
C A B A C B
D E E D

Byrne and Johnson-Laird (1989) reported 61% and 50% correct responses in one- and two-state problems respectively. Similarly, Carreiras and Santamaria (1997) reported 99% and 89% correct responses in one- and two-state problems. There are also two-state problems that have no valid conclusion. Problem 3 results in two possible mental states contradicting each other. Problems with no valid conclusion result in the lowest proportion of correct responses. Two separate experiments by Byrne and Johnson-Laird resulted in 18% and 15% of correct responses in problems with no valid conclusions.

It is assumed that two-state problem is more difficult because it requires higher working memory load than one-state problem. However, it does not explain why accuracy drops even lower in a two-state problem with no valid conclusion. Our ACT-R model that uses HRM module provides a possible explanation for this effect.

Model's design The model's strategy can be divided into five steps:

1. The model constructs a mental state of the problem inside VSTM. The mental state is built iteratively by processing premises one by one and updating VSTM at each iteration. Items from a premise are converted into abstract visual objects and given *(x, y)* coordinates based on positions relative to the items already existing inside VSTM. A premise is also converted into a logical *statement* stored inside declarative memory, but it is done only after VSTM is updated. The model can handle two-state problems. For example, while processing the second premise in Problem 2, the model uses *assertion (r-dir-left-of "@item" B)* to check if there is already another item present to left of *B*. This assertion triggers bottom-up spatial reasoning and HRM returns any visual object that is to the left of *B*. In case of Problem 2, *A* is returned. Then the model stores both *C* and *A* inside its working memory as items to be swapped positions in a second mental state.
2. After all premises are processed and a mental state is built inside VSTM, the model sends assertion to HRM to try to answer a query. The assertion is in form of *("@property" D E)*. To answer the assertion, HRM uses bottom-up spatial reasoning.
3. If it is a one-state problem then the model does nothing else. However, if there are two possible mental states then, after answering the query, the model creates the second possible mental state inside VSTM. This is done by swapping positions of the two objects previously stored inside working memory. In case of Problem 2, *C* is placed at the position of *A*, and *A* is placed at the position of *C*.
4. At this step, the model checks if any visual object was positioned relative to the swapped objects. If that is the case then the model verifies if relations still hold, if not then positions of those objects are corrected as well.
5. After creating the second mental state, the model sends to HRM the same assertion as in step 2. The answer for this assertion is compared to the answer from step 2. If answers are not the same then the model assumes that problem does not have valid conclusion.

Results Model's proportions of correct responses in one-state problems, two-state problems with valid conclusion and two-state problems with no valid conclusion are 100%, 76% and 34% respectively. The model always gives correct answers in one-state problems. However, it starts making mistakes in two-state problems. Furthermore, the model shows lowest accuracy in two-state problems with no valid conclusion. The cause of mistakes is model's confusion between similar spatial properties such as *r-below* and *r-dir-below*.

The first mistake can be made during step 4. Consider following example from Problem 3 where the model just finished step 3 by swapping positions of *A* and *C*:

```
C A B      A C B
D E   ==>  D E
```

During step 4, the model has to verify whether the spatial relation between *D* and *C* still holds. One of two possible assertions can be used for such verification: *(r-below D C)* or *(r-dir-below D C)*. The model's choice is random in this case. However, if *r-below* is used then the assertion will be evaluated to be true since bottom-up reasoning with *r-below* does not check for vertical alignment. This leads the model to a wrong conclusion that *D*'s position does not need to be corrected. Such mistake can lead to a situation where, for example, in problem 3, the relation between *D* and *E* is still the same in both mental states. The second mistake can be made during comparison in step 5. Let us consider the case where, in problem 2, the answers to the assertions in step 2 and 5 were *(r-left-of D E)* and *(r-dir-left-of D E)* respectively. These two statements, although similar, are not the same. Hence, if no explicit top-down reasoning is used to prove that one entails the other, the two answers are considered different. The model decides randomly whether to invoke top-down verification since it is not always necessary.

The model makes more mistakes in two-state problems with no valid conclusion because it is vulnerable to both types of mistakes in those problems. However, only second mistake is possible in two-state problems with valid conclusion. In one-state problems, the model is not susceptible to any of those mistakes.

Discussion and Conclusion

The current implementation of HRM is very much at a prototype phase, and its features may change with future revisions. However, preliminary results from the models of casual deduction and spatial relations tasks are promising.

The casual deduction model is a demonstration of how a triple-based knowledge structure can help to explain how complex background knowledge can influence an outcome of even simple deductive reasoning. As such, it is no longer a deductive reasoning, but rather a pragmatic reasoning, a reasoning based on both a given propositional form and its content, previous knowledge (Braine & O'Brien, 1991). It is interesting to see a rise of the pragmatic reasoning in HRM since it does not incorporate any dedicated controls for it. The very dependency of HRM's deductive reasoning on ACT-R's declarative mechanisms gives rise to a quite natural pragmatic reasoning. As such, there is a possibility that a pragmatic reasoning is not a different logical process, but a deductive reasoning bound by properties and limitations of our long-term declarative memory.

The model of spatial relations task shows an in depth view of how rule- and mental model-based reasoning strategies are used together in the same task. It is imperative for a success that both strategies complement each other. The core of model-based reasoning is bottom-up reasoning, an ability to derive explicit knowledge from an implicit knowledge. Although fast and efficient, bottom-up reasoning has limitations on the complexity of semantics it can operate. Those limitations make the model-based reasoning prone to mistakes if not corrected by rule-based reasoning. In the other hand, top-down rule-based reasoning is a slow and costly process not feasible for real-time interactive tasks. It has to rely on a model-based reasoning to speed up the reasoning process. When a reasoning pipeline recursively calls itself, it blurs the boundary between rule- and model-based strategies since both of them may be used during the same reasoning process.

As a framework, HRM provides a unified view of two reasoning strategies. Just like human mental logic, HRM, as a module, was designed to be general and task-independent. These properties make HRM suitable for modeling wide variety of tasks. The source code and related data for HRM module and validation models can be downloaded from here: http://www.ai.rug.nl/~n_egii/models/.

References

Anderson, J. R. (2007). *How Can Human Mind Occur in the Physical Universe?* New York: Oxford University Press.

Braine, M. D., & O'Brien, D. P. (1991). A Theory of "If": A Lexical Entry, Reasoning Program, and Pragmatic Principles. *Psychological Review , 98* (2), 182-203.

Byrne, R. M., & Johnson-Laird, P. N. (1989). Spatial Reasoning. *Journal of Memory and Language , 28*, 564-575.

Carreiras, M., & Santamaria, C. (1997). Reasoning About Relations: Spatial and Nonspatial Problems. *Thinking and Reasoning , 3* (3), 191-208.

Crystal, D. (1997). *The Cambridge Encyclopedia of Language.* Cambridge: Cambridge University Press.

Cummins, D. D. (1995). Naive theories and causal deduction. *Memory & Cognition , 23* (5), 646-658.

Cummins, D. D., Lubart, T., Alksnis, O., & Rist, R. (1991). Conditional reasoning and causation. *Memory & Cognition , 19* (3), 274-282.

Johnson-Laird, P. N. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness.* Cambridge: Cambridge University Press.

Nyamsuren, E., & Taatgen, N. A. (2013). Pre-attentive and Attentive Vision Module. *Cognitive Systems Research , 24*, 62-71.

Rensink, R. A. (2007). The modeling and control of visual perception. In W. D. Gray, *Integrated Models of Cognitive Systems* (pp. 132-148). New York: Oxford.

Rips, L. J. (1983). Cognitive processes in propositional reasoning. *Psychological Review , 90* (1), 38-71.

Roberts, M. J. (1993). Human Reasoning: Deduction Rules or Mental Models, or Both? *The Quarterly Journal of Experimental Psychology , 46A* (4), 569-589.

Wintermute, S. (2012). Imagery in cognitive architecture: Representation and control at multiple levels of abstraction. *Cognitive Systems Research , 19-20*, 1-29.