# Behaviour and Reasoning Description Language (BRDL)

Antonio Cerone(✉)

Department of Computer Science, Nazarbayev University, Nur-Sultan, Kazakhstan
antonio.cerone@nu.edu.kz

**Abstract.** In this paper we present a basic language for describing human behaviour and reasoning and present the cognitive architecture underlying the semantics of the language. The language is illustrated through a number of examples showing its ability to model human reasoning, problem solving, deliberate behaviour and automatic behaviour. We expect that the simple notation and its intuitive semantics may address the needs of practitioners from non matematical backgrounds, in particular psychologists, linguists and other social scientists. The language usage is twofold, aiming at the formal modelling and analysis of interactive systems and the comparison and validation of alternative models of memory and cognition.

**Keywords:** Cognitive science · Human reasoning · Problem solving · Human behavior · Formal methods

## 1 Introduction

Research in modelling human cognition has resulted in the development of a large number of *cognitive architectures* over the last decades [9,17]. However, we are still very far from having a unified approach to modelling cognition. In fact, cognitive architectures are based on three different modelling approaches, *symbolic* (or *cognitivist*), such as Soar [10], which are based on a set of predefined general rules to manipulate symbols, *connectionist* (or *emergent*), such as DAC [19], which count on emergent properties of connected processing components (e.g. nodes of a neural network), and *hybrid*, such as CLARION [18], which combine the two previous approaches. Moreover, there is no clear agreement on the categorisation of specific architecture in this taxonomy. For example, ACT-R [1] is often classified as symbolic but, in fact, explicitly self-identifies as hybrid. Furthermore, most architectures have been developped for research purposes and are fairly specialised in one or more of the following areas: psychological experiments, cognitive robotics, human performance modelling, human-robot interaction, human-computer interaction, natural language processing, categorisation and clustering, computer vision games and puzzles, and virtual agents [9].

The complexity of these cognitive architectures makes it difficult to fully understand their semantics and requires high expertise in programming them.


J. Camara and M. Steffen (Eds.): SEFM 2019 Workshops, LNCS 12226, pp. 137–153, 2020.
https://doi.org/10.1007/978-3-030-57506-9_11

Moreover, although cognitive architectures can mimic many aspects of human behaviour and learning, they never really managed to be easily incorporated in the system and software verification process.

In this paper we propose a notation, the *Behaviour and Reasoning Description Language (BRDL)*, for describing human behaviour and reasoning. The semantics of the language is based on a basic model of human memory and memory processes and is adaptable to different cognitive theories. This allows us, on the one hand, to keep the syntax of the language to a minimum, thus making it easy to learn and understand and, on the other hand, to use alternative semantic variations to compare alternative theories of memory and cognition. The latter can be easily achieved by replacing implementation modules and, on a finer grain, varying the values of a number of semantic parameters.

BRDL originated from and extends the *Human Behaviour Description Language (HBDL)* introduced in our previous work [2,3]. HBDL focuses on the modelling of automatic and deliberate behaviour. However, it requires reasoning and problem solving aspects to be modelled explicitly in a procedural way, whereby the reasoning process and the problem solution are explicitly described with the language. BRDL, instead, is equipped with the linguistic constructs to specify reasoning goals (e.g. questions), inference rules and unsolved problems. The cognitive engine implementing the language then emulates the reasoning and problem solving processes. In our previous work [2,3], HBDL has been implemented using the Maude rewrite language and system [11,16]. In our recent work [4] we started implementing BRDL using the real-time extension of Maude [15]. The use of formal methods, specifically Maude, to implement the languages allows us to combine human components and system components and perform formal verification. This is carried out by exploiting the model checking capability of Maude and Real-time Maude.

This paper aims at addressing a broad community of researchers from different backgrounds but all interested in cognition. For this reason, rather than listing formal definitions, we start from small, practical examples and then generalise them as semi-formal definitions or algorithmic descriptions in which we avoid jargon and keep the formal notation to a minimum. Formality is introduced, usually in term of elementary set theory, only when is needed to avoid ambiguity, but is avoided whenever a textual explanation is sufficient.

Section 2 introduces the underlying memory and cognitive model, inspired by the information processing approach. Section 3 describes the notation used for knowledge representation and presents the algorithm used for knowledge retrieval. Section 4 presents how to model deliberate behaviour in term of reasoning, interaction and problem solving. In particular, it illustrates how inference rule are used in reasoning and interaction and how knowledge drives the decomposition of the problem goal into subgoals. Section 5 presents how to model automatic behaviour and how this evolves from deliberate behaviour through skill acquisition. Finally, Sect. 6 concludes the paper and discusses the ongoing BRDL implementation as well as future work.

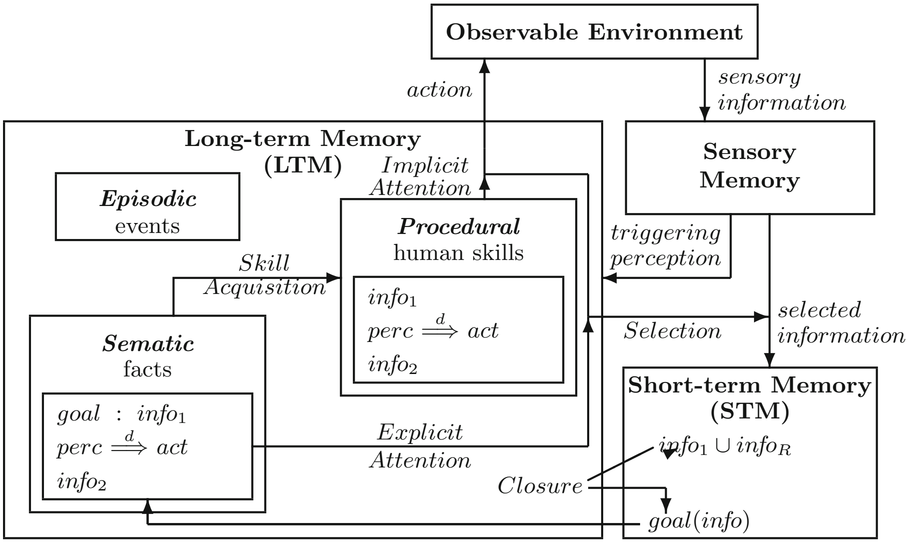


**Fig. 1.** Human memory architecture underlying BRDL semantics

## 2 Human Memory Architecture

Following the *information processing* approach normally used in cognitive psychology, we model human cognitive processes as processing activities that make use of input-output channels, to interact with the external environment, and three main kinds of memory, to store information. Input and output occur through the senses and the motor system. We give a general representation of input channels in term of *sensory information*, possibly abstracting away from the specific senses that are used. We represent output channels in term of *actions* performed on the observable environment.

Figure 1 describes the human memory architecture we will use to provide the semantics of BRDL. The notational details of the figure will be explained in Sects. 4 and 5. The memory consists of the following components:

***sensory memory***
: where information perceived through the senses persists for a very short time [13];

***short-term memory (STM)***
: which has a limited capacity and where the information that is needed for processing activities is temporarily stored with rapid access and rapid decay [6,7,13];

***long-term memory (LTM)***
: which has a virtually unlimited capacity and where information is organised in structured ways, with slow access but little or no decay [5,8].

We must note that the term STM indicates a mere, short-term storage of information, whereas the term *working memory* is used for a short-term buffer that also supports processing and manipulation of information [6,7]. Although some neuropsychological studies show evidences supporting this distinction, which correspond to two different neural subsystems within the prefrontal cortex [7], in our work we do not associate processing with memory directly. In fact, we consider the short-term storage aspects as a whole and express them in the BRDL syntax, while all processing aspects are delegated to the semantics of the language.

As shown in Fig. 1, we consider a human memory architecture in which, depending on the content of the LTM, some *perception* (*perc*) selected among the *sensory information* stored in sensory memory, in combination with information ($info_1$) and possibly a goal (*goal*) stored in STM, *triggers* some human *action* (*act*) on the observable environment and/or the transfer of the (possibly processed) selected information from sensory memory to STM ($info_2$).

A usual practice to keep information in memory is *rehearsal*. In particular, *maintenance rehearsal* allows us to extend the time during which information is kept in STM, whereas *elaborative rehearsal* allows us to transfer information from STM to LTM.

### 2.1 Short-Term Memory (STM) Model

The limited capacity of the STM has been measured using experiments in which the subjects had to recall items presented in sequence. By presenting sequences of digits, Miller [12] found that the average person can remember $7 \pm 2$ digits. However, when digits are grouped in *chunks*, as it happens when we memorise phone numbers, it is actually possible to remember larger numbers of digits. Therefore, Miller's $7 \pm 2$ rule applies to chunks of information and the ability to form chunks can increase people's STM actual capacity.

We assume that the STM may contain pieces of information, which may describe cognitive information, possibly retrieved from the LTM, goals, recent perceptions or planned actions. Therefore we can denote the set of pieces of information that may be in STM as

$$\Theta = \Pi \cup \Sigma \cup \Delta \cup \Gamma,$$

where $\Pi$ is a set of perceptions, $\Sigma$ is a set of mental representations of human actions, $\Delta$ is a set of pieces of cognitive information and $\Gamma$ is a set of goals. Moreover, each piece of information is associated with a *life time*, which is initialised as the *STM decay time* when the information is first stored in the STM and then decremented as time passes. A piece of information disappears from the STM once its life time has decreased to 0.

The limited capacity of short-term memory requires the presence of a mechanism to empty it when the stored information is no longer needed. When we produce a chunk, the information concerning the chunk components is removed from the STM. For example, when we chunk digits, only the representation of the chunk stays in the STM, while the component digits are removed and can no longer be directly remembered as separate digits. Generally, every time a task is completed, there may be a subconscious removal of information from STM, a process called *closure*: the information used to complete the task is likely to be removed from the STM, since it is no longer needed. Therefore, when closure occurs, a piece of information may disappear from the STM even before its life time has decrease to 0. Furthermore, a piece of information may disappear from the STM also when the STM has reached its maximum capacity and it is needed to make space for the storage of needed information. Conversely, *maintenance rehearsal* resets the life time to the value of the decay time.

### 2.2 Long-Term Memory (LTM) Model

Long term memory is divided into two types

***declarative*** **or** ***explicit*** **memory**
: refers to our knowledge of the world ("knowing what") and consists of the *events* and *facts* that can be *consciously* recalled:
  - our experiences and specific events in time stored in a serial form (*episodic memory*);
  - structured record of facts, meanings, concepts and knowledge about the external world, which we have acquired and organised through association and abstraction (*semantic memory*).

***procedural*** **or** ***implicit*** **memory**
: refers to our skills ("knowing how") and consists of *rules* and *procedures* that we *unconsciously* use to carry out tasks, particularly at the motor level.

Emotions and specific contexts and environments are factors that affect the storage of experiences and events in episodic memory. Information can be transferred from episodic to semantic memory by making abstractions and building associations, whereas *elaborative rehearsal* facilitates the transfer of information from STM to semantic memory in an organised form.

Note that also declarative memory can be used to carry out tasks, but in a very inefficient way, which requires a large mental effort in using the STM (*high cognitive load*) and a consequent high energy consumption. In fact, declarative memory is heavily used while learning new skills. For example, while we are learning to drive, ride a bike, play a musical instrument or even when we are learning to do apparently trivial things, such as tying a shoelace, we consciously retrieve a large number of facts from the semantic memory and store a lot of information in the STM. Skill acquisition typically occurs through repetition and practice and consists in the creation in the procedural memory of rules and procedures (*proceduralisation*), which can be then unconsciously used in an automatic way with limited involvement of declarative memory and STM.

### 2.3 Memory Processes and Cognitive Control

We have mentioned in Sect. 2.2 that skill acquisition results in the creation in procedural memory of the appropriate rules to automatically perform the task, thus reducing the accesses to declarative memory and the use of STM, and, as a result, optimising the task performance.

As shown in Fig. 1, sensory information is briefly stored in the sensory memory and only relevant information is transferred, possibly after some kind of processing, to the STM using *attention*, a selective processing activity that aims to focus on one aspect of the environment while ignoring others. *Explicit attention* is associated with our goal in performing a task and is activated by the content of the semantic memory. It focusses on goal-relevant stimuli in the environment. *Implicit attention* is grabbed by sudden stimuli that are associated with the current mental state or carry emotional significance. It is activated by the content of the procedural memory.

Inspired by Norman and Shallice [14], we consider two levels of cognitive control:

***automatic control***
fast processing activity that requires only *implicit attention* and is carried out outside awareness with no conscious effort implicitly, using rules and procedures stored in the procedural memory;

***deliberate control***
processing activity triggered and focussed by *explicit attention* and carried out under the intentional control of the individual, who makes explicit use of facts and experiences stored in the declarative memory and is aware and conscious of the effort required in doing so.

For example, automatic control is essential in properly driving a car and, in such a context, it develops throughout a learning process based on deliberate control. During the learning process the driver has to make a conscious effort that requires explicit attention to use gear, indicators, etc. in the right way (deliberate control). In fact, the driver would not be able to carry out such an effort while talking or listening to the radio, since the deliberate control is entirely devoted to the driving task. Once automaticity in driving is acquired, the driver is no longer aware of low-level details and resorts to implicit attention to perform them (automatic control), while deliberate control and explicit attention may be devoted to other tasks such as talking or listening to the radio.

One of the uses of BRDL is the analysis and comparison of different architectural models of human memory and cognitions. In this sense the semantics of the language depends on the values assigned to a number of parameters, such as:

***STM maximum capacity*** the maximum number of pieces of information (possibly chuncks) that can be stored in STM;

***STM decay time*** the maximum time that information may persist in STM in absence of maintenance rehearsal;

***lower closure threshold*** the minimum STM load to enable closure;

***upper closure threshold*** the minimum STM load to force closure;

***LTM retrieval maximum time*** the maximum time that can be used to retrieve information from LTM before a retrieval failure occurs.

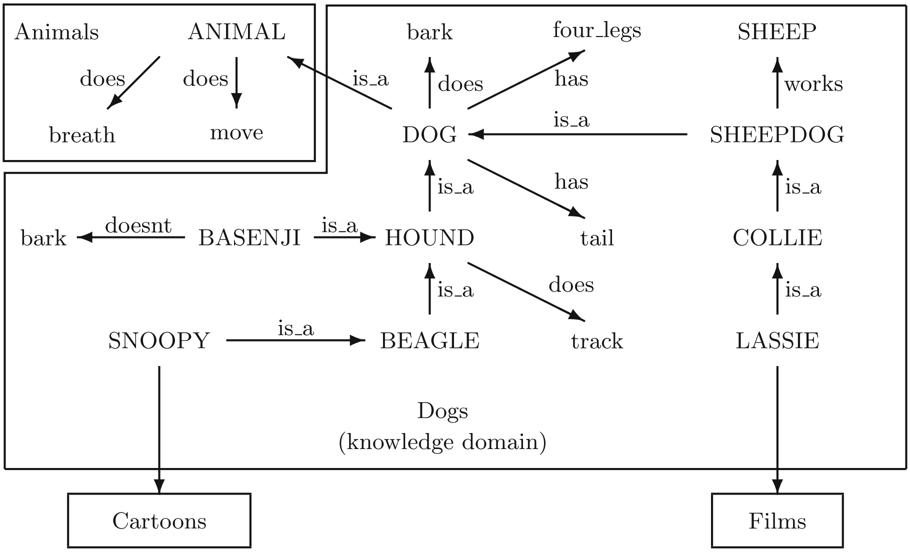


**Fig. 2.** Example of semantic network (adapted from Dix' work [8]).

## 3 Knowledge Representation

*Semantic networks* are a simple, effective way to describe how we represent and structure information in semantic memory. We call *category* any item that is the object of our knowledge. An association between two categories is described by a labelled arrow. A label is used to specify the nature of the association. For example, an arrow labelled with "is_a" denotes a *generalisation*: the arrow goes from the more specific to the more generic category. Additionally, a category may have *attributes*, which may also be categories. The arrow from the category to one of its attribute is labelled with a *type* characterising the relationship between the attribute and the category. In the case of generalisation, the more specific category inherits all attributes of the more generic category unless the attribute is redefined at the more specific category level.

For example, Fig. 2 shows a semantic network for the *knowledge domain dogs*. Note that in Fig. 2 we have used words entirely in upper-case letters for categories and capitalised words for knowledge domains for readability purposes. However, this convention is not used in BRDL. Note that the *bark* attribute is duplicated only for better readability of the semantic network. In fact, a single *bark* attribute should occur as the target of two arrows, one with source *dog* and label *does* and on with source *basenji* and label *doesnt*.

Associations are described in BRDL either in terms of the application of the label *is_a* to a category (*generalisation*) or in terms of the application of a type to an attribute (*typed attribute*). For example, the *dog* category is generalised as the more generic category *animal* ($is\_a(animal)$) at a higher level and has the following typed attributes with obvious meaning: $does(bark)$, $has(four\_legs)$

and $has(tail)$. Category *dog* is also the $is\_a(dog)$ generalisation of lower-level categories describing dog groups, such as *sheepdog* and *hound*, which are in turn generalisations of even lower-level categories describing dog breeds, such as *collie*, *beagle* and *basenji*. Furthermore, category *basenji* has the $doesnt(bark)$ typed attribute, which redefines the $does(bark)$ typed attribute of the *dog* category. In fact, a basenji is an exceptional dog breed that does not bark.

A *fact representation* in semantic memory is modelled in BRDL as

$$domain : category \mid \xrightarrow{delay} \mid type(attribute)$$

where *delay* is the mental processing time needed to retrieve the association between category *category* and type attribute $type(attribute)$ within the given knowledge domain *domain*. With reference to Fig. 2, obvious examples of fact representations are:

1. $animals : animal \mid \xrightarrow{d_1} \mid does(breath)$,
2. $animals : animal \mid \xrightarrow{d_2} \mid does(move)$,
3. $dogs : dog \mid \xrightarrow{d_3} \mid is\_a(animal)$,
4. $dogs : dog \mid \xrightarrow{d_4} \mid does(bark)$,
5. $dogs : hound \mid \xrightarrow{d_5} \mid is\_a(dog)$,
6. $dogs : basenji \mid \xrightarrow{d_6} \mid is\_a(hound)$,
7. $dogs : hound \mid \xrightarrow{d_7} \mid does(track)$,
8. $dogs : basenji \mid \xrightarrow{d_8} \mid doesnt(bark)$ .

There are some relations between attribute types. For instance, *doesnt* is the negation of *does* and $isnt\_a$ is the negation of $is\_a$.

### 3.1 Knowledge Retrieval

Knowledge retrieval occurs deliberately, driven by specific goals we have in mind. Our working memory is the STM, so our current goals are stored in STM. Within a given knowledge *domain*, we model the goal of retrieving the *attributes* of a given *type* that are associated with a given *category* as

$$goal(domain, type\_what?(category)).$$

The presence of such a goal in STM triggers the retrieval of one specific *attribute* so that a new piece of *cognitive information*, either fact $type(category, attribute)$ or its negation is added to the STM, unless it is already there, while the goal is removed from the STM. If the fact is already in STM, then other attributes will be retrieved until we have a fact that is not in STM yet, and can thus added to it. If there are more facts matching the goal that are not in STM yet, then the one whose representation in LTM has the least mental processing time is retrieved. One of the memory parameters introduced in Sect. 2.3, the *LTM* *retrieval maximum time*, defines the maximum time for such a search, after which the $dontknow(domain, type\_what?(category))$ fact replaces the goal in STM.

Suppose that we want to find out what an animal does. Our goal is

$$goal(animals, does\_what?(animal)).$$

This goal immediately matches fact representations 1 and 2 in LTM introduced in the example in Sect. 3. Thus the goal is replaced in STM by $does(animal, breath)$ after time $d_1$, if $d_1 < d_2$, or by $does(animal, move)$ after time $d_2$, if $d_2 < d_1$. If $d_1 = d_2$, then the choice is nondeterministic.

Other possible goals are:

$goal(domain, type\_which?(attribute))$ for retrieving the *category* with which $type(attribute)$ is associated;

$goal(domain, type?(category, attribute))$ for answering the question on whether *category* is associated with $type(attribute)$ and, if the answer is positive, adding fact $type(category, attribute)$ to the STM, otherwise adding its negation to the STM.

We want now to find out whether a basenji breaths. Our goal is

$$goal(dogs, does?(basenji, breath)).$$

Since none of the attributes of *basenji* matches our question we need to climb the hierarchy of categories described in Sect. 3 and go through *hound* (fact representation 6) and *dog* (fact representation 5) until we reach *animal* (fact representation 3) and find out that our question matches fact representation 1. The time for such a retrieval is the sum of the retrieval times of all *is_a* fact representations for all categories we have gone through ($d_6$, $d_5$ and $d_3$) plus the sum of the retrieval times of all *does* facts associated with each of these categories that do not match the goal ($d_7$) plus the retrieval time of the fact representation that matches the goal ($d_1$): $d_6 + d_5 + d_3 + d_7 + d_1$, which is obviously greater than time $d_1$ needed to find out whether an animal breaths. This is consistent with Collins and Quillian's experiments on retrieval time from semantic memory [5].

Finally, we want to find out whether a basenji barks. Our goal is

$$goal(dogs, does?(basenji, bark)).$$

This goal immediately matches fact representation 8 in LTM introduced in the example in Sect. 3. Thus the goal is replaced in STM by $doesnt(basenjil, bark)$ after time $d_7$.

In general, given goal $g(c) = goal(dom, type\_what?(c))$ and an LTM retrieval maximum time $d_{max}$, the fact $f(g, c)$ that replaces the goal in STM after time $t(g, c)$ is defined as follows:

1. $f(g, c) = type(c, a)$ with $t(g, c) = d$
   if $dom : c \mid \xrightarrow{d} \mid type(a)$ is in LTM;

2. $f(g,c) = f(g,c')$ with $t(g,c) = s(type,c) + d' + t(g,c')$
   if there is no attribute $a$ such that $dom : c \mid \xrightarrow{d} \mid type(a)$ is in LTM and $t(g,c) < d_{max}$ and there is a knowledge domain $dom'$ such that $dom' : c \mid \xrightarrow{d'} \mid is\_a(c')$ is in LTM;
3. $f(g,c) = \overline{type}(c,a)$ with $t(g,c) = d_{max}$
   if there is no fact in LTM that can be retrieved within time $d_{max}$

where $\overline{type}$ is the negation of *type* and $s(type,c)$ is the sum of the retrieval times of all fact representations in LTM with the given category $c$ and type *type*. The attribute $a$ associated with category $c$ may be retrived without climbing the hierarchy of categories (1) or may be required to climb the hierarchy (iteration of 2) or may not be found at all (3).

Similar algorithms can be given for goals $goal(dom, type\_which?(a))$ and $goal(dom, type?(c,a))$. Note that $goal(dogs, does?(basenji, bark))$ would retrieve the fact $does(bark, dog)$ without considering the exception *basenji*, which is at a lower level than *dog* in the hierarchy. This is consistent with the fact that we normally neglect exceptions when we make general cosiderations.

We conclude this section with a clarification about the role of the knowledge domain. Although retrieval goes across knowledge domains, it is the existence of a specific knowledge domain to enable it. For example, with reference to Fig. 2, knowledge domain 'Dogs' allows us to retrieve information on 'SNOOPY' as a dog, but not as a cartoon character. That is, we can find out that SNOOPY tracks but not that SNOOPY thinks. This last piece of information, instead, could be retrieved within the 'Cartoon' knowledge domain.

## 4 Deliberate Basic Activities

Fact representations in semantic memory not only describe the *static knowledge* of the world but also the *dynamic knowledge* on how to deliberately manipulate our own internal knowledge and understand the external world (*reasoning* and *problem solving*) and how to use knowledge to perceive and manipulate the external world (*interaction* and *problem solving*).

The general structure of a deliberate basic activity is

$$goal : info_1 \uparrow perc \stackrel{d}{\Longrightarrow} act \downarrow info_2$$

where

- $goal \in \Gamma$ is a goal, which may be structured in different ways;
- $perc \in \Pi$ is a perception on which the human *explicitly* focusses;
- $info_1 \subseteq \Theta \backslash \Gamma$ is the information retrieved and removed from the STM;
- $info_2 \subseteq \Theta$ is the information stored in the STM;
- $act \in \Sigma$ is the mental representation of a human action;
- $d$ is the mental processing time (up to the moment action *act* starts, but not including *act* duration).

The upward arrow denotes that $info_1$ is removed from the STM and the downward arrow denotes that $info_2$ is added to the STM. In case $info_1$ must not be removed from the STM we can use the following derived notation:

$$goal : info_1 \,|\, perc \stackrel{d}{\Longrightarrow} act \downarrow info_2$$

where the '|' instead of '↑' denotes that $info_1$ is not removed from the STM. This derived notation is equivalent to $goal : info_1 \uparrow perc \stackrel{d}{\Longrightarrow} act \downarrow info_1 \cup info_2$. Special cases are:

$goal : info_1 \uparrow \stackrel{d}{\Longrightarrow} act \downarrow info_2$ and $goal : info_1 \,|\, \stackrel{d}{\Longrightarrow} act \downarrow info_2$
if there is no perception;
$goal : info_1 \uparrow perc \stackrel{d}{\Longrightarrow} \downarrow info_2$ and $goal : info_1 \,|\, perc \stackrel{d}{\Longrightarrow} \downarrow info_2$
if there is no action;
$goal : info_1 \uparrow \stackrel{d}{\Longrightarrow} \downarrow info_2$ and $goal : info_1 \,|\, \stackrel{d}{\Longrightarrow} \downarrow info_2$
if there is neither perception nor action.

### 4.1 Goals

We have seen in Sect. 3.1 that a goal $goal(dom, q)$ for knowledge retrieval means that we deliberately look for an answer to question $q$ within knowledge domain $dom$. Once the answer is found or the ignorance of the answer is established, the goal is achieved and is removed from STM.

In more complex deliberate activities the knowledge domain might be related to the underlying *purpose* in our behaviour or represent a specific *task* to carry out. Thus goal $goal(dom, info)$ means that we deliberately want to achieve the information given by a non-empty set $info \subseteq \Theta \backslash \Gamma$, which may comprise one experienced perception, one performed action and some of the information stored in the STM except goals. Therefore, a goal $goal(dom, info)$ in STM is achieved when

- the human experiences $perc \in info$ or $\Pi \cap info = \emptyset$, and
- the human performs $act \in info$ or $\Sigma \cap info = \emptyset$, and
- $info \backslash \Pi \backslash \Sigma$ is included in STM,

where set difference '\' is left associative.

### 4.2 Reasoning

One way to manipulate our internal knowledge is to infer new facts from other facts whose representations are in our LTM. The inferred facts are added to the STM and may be preserved for the future either by transferring them to LTM through elaborative rehearsal or by recording them in the external environment in some way, e.g. through writing.

The LTM contains inference rules that we have learned throughout our life and are applied deliberately. For example, consider a person who is learning to drive. At some point throughout the learning process, the person learns the following rule:

*A driver has to give way to pedestrians ready to walk across the road on a zebra crossing.*

The premises of this rule are

*zebra*—there is a zebra crossing, and
*ped*—there are pedestrians ready to walk across the road.

The consequences is

$goal(driving, gw)$—the driver's goal is to give way to the pedestrians,

where $gw$ is the fact that the driver has given way to the pedestrians, which has to be achieved.

*Inference rule*

$$infer(driving) : \{zebra, ped\} \mid \stackrel{d}{\Longrightarrow}\downarrow \{goal(driving, gw)\},$$

models the fact that from the set of premises $\{zebra, ped\}$ we can infer the set of consequences $\{goal(driving, gw)\}$ in knowledge domain *driving*. The premises are not removed from the STM after applying the inference.

The general structure of an inference rule is

$$infer(dom) : premises \uparrow\stackrel{d}{\Longrightarrow}\downarrow consequences.$$

The rule is enabled when special goal *infer* and the *premises* are in STM. The application of the rule requires time $d$ and removes both special goal *infer* and the *premises* from STM and add the *consequences* to it. Since normally premises are not removed after applying the inference, it is common to use derived rule

$$infer(dom) : premises \mid \stackrel{d}{\Longrightarrow}\downarrow consequences,$$

which is equivalent to $infer(dom) : premises \uparrow\stackrel{d}{\Longrightarrow}\downarrow premises \cup consequences$.

Reasoning inference rules support all three main human reasoning modes: *deduction*, *abduction* and *induction*. The rule for giving way to pedestrian presented above is an example of *deduction*.

The following example of *abduction*

*A train that does not arrive at the scheduled time is late.*

can be modelled as

$$infer(railway) : \{arrivalTimePassed, noTrain\} \mid \stackrel{d}{\Longrightarrow}\downarrow \{trainLate\}.$$

In this case the inference goes from the *events*, i.e. the arrival time is passed and the train has not arrived yet, to the *cause*, i.e. the train is late. In reality, the train might have been cancelled rather than being late.

Finally, the following example of *induction* or *generalisation*

*if three trains in a row arrive late then all trains arrive late.*

can be modelled as

$$infer(railway) : \{train1Late, train2Late, train3Late\} \mid \stackrel{d}{\Longrightarrow}\downarrow \{allTrainsLate\}.$$

### 4.3 Interaction

Interaction concerns the perception and the manipulation of the external world making use of internal knowledge. Consider again a person who is learning to drive and has to deal with a zebra crossing. Normally the explicit attention of a learner who is driving a car tries to focus on a large number of perceptions. If we restrict the driving task (*driving*) to just a zebra crossing, explicit attention involves only two perceptions, *zebra* and *ped*, and is driven by two goals, $driving, \{zebra\}$ and $driving, \{ped\}$, which are simultaneously in STM.

This restricted driving task may be modelled in BRDL as:

1. $goal(driving, \{zebra\}) : \emptyset \,|\, zebra \overset{d_1}{\Longrightarrow}\downarrow \{zebra\}$,
2. $goal(driving, \{ped\}) : \emptyset \,|\, ped \overset{d_2}{\Longrightarrow}\downarrow \{ped\}$,
3. $goal(driving, \{ped\}) : \{zebra\} \,|\, ped \overset{d_3}{\Longrightarrow}\downarrow \{ped, infer(driving)\}$,
4. $goal(driving, \{zebra\}) : \{ped\} \,|\, zebra \overset{d_4}{\Longrightarrow}\downarrow \{zebra, infer(driving)\}$,
5. $goal(driving, \{gw\}) : \emptyset \,|\, \overset{d_5}{\Longrightarrow} stop \downarrow \{gw\}$.

After the driver has perceived the presence of zebra crossing and pedestrians and stored *zebra* and *perc* in the STM (basic activities 1 and 3 or 2 and 4), an inference rule enabled by the content of the STM is searched. This is the rule defined in Sect. 4.2, which store *gw* in the STM, thus informing the driver about the need to give way to the pedestrian. The driver complies with the rule by performing action *stop* to stop the car (basic activity 5).

### 4.4 Problem Solving

Problem solving is the process of finding a solution to an unfamiliar task. In BRDL problems to be solved are modelled by goals stored in STM. We illustrate with an example how the knowledge stored in LTM may lead to the solution.

Consider the task of moving a box full of items. The STM contains

- goal $goal(boxes, \{moved, full\})$;
- pieces of information $notMoved$ and $full$.

Suppose to have the following obvious knowledge stored in LTM:

1. $goal(boxes, \{full\}) : \,|\, full \overset{d_1}{\Longrightarrow}\downarrow \{full\}$
2. $goal(boxes, \{empty\}) : \,|\, empty \overset{d_2}{\Longrightarrow}\downarrow \{empty\}$
3. $goal(boxes, \{moved\}) : \{empty, notMoved\} \uparrow\overset{d_3}{\Longrightarrow} move \downarrow \{empty, moved\}$
4. $goal(boxes, \{empty\}) : \{full\} \uparrow\overset{d_4}{\Longrightarrow} remove \downarrow \{empty\}$
5. $goal(boxes, \{full\}) : \{empty\} \uparrow\overset{d_5}{\Longrightarrow} fill \downarrow \{full\}$

Basic activities 1 and 2 model the explicit attention on whether the box is full or empty. Basic activities 3 models the moving of an empty box. Basic activities 4 models the filling of an empty box. Basic activities 5 models the removal of all items from a full box. We assume that the box may be filled or emptied with just a single action.

None of the basic activities in LTM is enabled by the contents of the STM. Therefore, first goal $goal(boxes, \{moved, full\})$ is decomposed into two goals of knowledge domain *boxes* that control basic activities in LTM

$$goal(boxes, \{moved\}) \text{ and } goal(boxes, \{full\})$$

and is replaced by them after time $d_1 + d_2 + d_3 + d_4 + d_5$, which is needed to explore all basic activities within the knowledge domain. Then, the contents of the STM are removed from information $\{empty, notMoved\}$, which enables the basic activities that are controlled by the two goals but not triggered by perceptions. The resultant information $\{empty\}$ is what is missing from the STM to make progress in solving the problem. Therefore, a goal $goal(boxes, \{empty\})$ is added to the STM after a further $d_3 + d_5$ time.

Goal $goal(boxes, \{empty\})$ is considered first, since it is the last one that was added to the STM, and is achieved by performing basic activity 4. This makes the box empty, thus enabling basic activities 3 and 5. Between the two, basic activity 3 is chosen first since it is enabled by a larger amount of information ($\{empty, notMoved\}$ versus $\{empty\}$), thus moving the box and achieving goal $goal(boxes, \{moved\})$. Finally, basic activity 5 is performed and also goal $goal(boxes, \{full\})$ is achieved.

## 5 Automatic Basic Activities

Automatic basic activities are performed independently from the goals in the STM. The general structure of an automatic basic activity is

$$dom : info_1 \uparrow perc \overset{d}{\Longrightarrow} act \downarrow info_2$$

where

- *dom* is a knowledge domain, possibly a task;
- $perc \in \Pi$ is a perception on which the human *implicitly* focusses;
- $info_1 \subseteq \Theta \backslash \Gamma$ is the information retrieved and removed from the STM;
- $info_2 \subseteq \Theta$ is the information stored in the STM;
- $act \in \Sigma$ is the mental representation of a human action;
- $d$ is the mental processing time (up to the moment action *act* starts, but not including *act* duration).

Also for automatic basic activities, perception and/or action may be absent.

Automatic basic activities originate from the proceduralisation in procedural memory of repeatedly used deliberate activities in semantic memory. Consider the example of the behaviour of a driving learner at a zebra crossing, which was introduced in Sect. 4.3. After a lot of driving experience, the driver's behaviour will become automatic. From the five deliberate basic activity in semantic memory the following new automatic activity are created in procedural memory:

1. $driving : \emptyset \,|\, zebra \stackrel{d_1'}{\Longrightarrow} \downarrow \{zebra\}$,
2. $driving : \{zebra\} \,|\, ped \stackrel{d_2'}{\Longrightarrow} stop \downarrow \{ped\}$.

Automatic basic activity 1 models the skill driver's implicit attention focussing on the zebra crossing, whose presence is unconsciously noted while approaching it, either though direct sight or indirectly via a warning signal. With such an automatic behaviour, the mental processing time of a skilled drivers, who is aware of the presence of a zebra crossing, from the moment of the perception of the pedestrians to the moment the *stop* action starts is $d_2'$. Taking into account that the application of the zebra crossing inference rule introduced in Sect. 4.2 requires $d$ mental processing time, with the learner's deliberate behaviour modelled in Sect. 4.3 such a mental processing time is either $d_3 + d + d_5$, if the driver notices the zebra crossing first (deliberate basic activities 1 and 3), or $d_4 + d + d_5$, if the driver notices the pedestrians first (deliberate basic activities 2 and 4), which are both expected to be greater than $d_2'$. In this sense the skilled driver's behaviour is safer than the lerner's behaviour.

## 6 Conclusion and Future Work

We have introduced the Behaviour and Reasoning Description Language (BRDL) for describing human behaviour and reasoning as an extension of the Human Behaviour Description Language (HBDL) presented in our previous work [2,3]. BRDL semantics has been provided on-the-fly in terms of a basic model of human memory and memory processes. We are currently implementing BRDL [4] using Real-time Maude [15] as part of a formal modelling and analysis environment that includes both human components and system components [3].

The object-oriented nature of Real-time Maude supports a highly modular implementation with separate modules describing alternative theories of cognition. Moreover, the use of a number of parameters as the ones listed at the end of Sect. 2.3 supports a fine-grain control of the applicability of Maude rewrite rules. In our future work, we will use this feature to compare in-silico experiments that use different combinations of parameter values with the data collected from real-life observations and experiments. This is expected to provide a calibration of the cognitive architecture underlying BRDL and, hopefully, important insights into alternative cognitive theories.

Finally, BRDL is a basic language, easy to extend and adapt to new contexts. This important characteristic is matched at the implementation level by exploiting Maude equational logic to construct new, complex data types.

**Acknowledgments.** The author would like to thank the four anonymous reviewers whose comments and suggestions greatly contributed to improve the paper.

## References

1. Anderson, J.R.: The Architecture of Cognition. Psychology Press, East Sussex (1983)

2. Cerone, A.: A cognitive framework based on rewriting logic for the analysis of interactive systems. In: De Nicola, R., Kühn, E. (eds.) SEFM 2016. LNCS, vol. 9763, pp. 287–303. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41591-8_20
3. Cerone, A.: Towards a cognitive architecture for the formal analysis of human behaviour and learning. In: Mazzara, M., Ober, I., Salaün, G. (eds.) STAF 2018. LNCS, vol. 11176, pp. 216–232. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04771-9_17
4. Cerone, A., Ölveczky, P.: Modelling human reasoning in practical behavioural contexts using Real-time Maude. In: FM Collocated Workshops 2018 (FMIS). Lecture Notes in Computer Science, vol. 12025. Springer (2019). https://doi.org/10.1007/978-3-030-54994-7_32, In press
5. Collins, A.M., Quillian, M.R.: Retrieval time from semantic memory. J. Verbal Learn. Verbal Behav. **8**, 240–247 (1969)
6. Cowan, N.: What are the differences between long-term, short-term, and working memory? Prog. Brain Res. **169**, 223–238 (2008)
7. Diamond, A.: Executive functions. Annu. Rev. Psychol. **64**, 135–168 (2013)
8. Dix, A., Finlay, J., Abowd, G., Beale, R.: Human-Computer Interaction, 3rd edn. Pearson Education, London (2004)
9. Kotseruba, I., Tsotsos, J.K.: 40 years of cognitive architectures: core cognitive abilities and practical applications. Artif. Intell. Rev. **53**(1), 17–94 (2018). https://doi.org/10.1007/s10462-018-9646-y
10. Laird, J.A.: The Soar Cognitive Architecture. MIT Press, Cambridge (2012)
11. Martí-Oliet, N., Meseguer, J.: Rewriting logic: roadmap and bibliography. Theor. Comput. Sci. **285**(2), 121–154 (2002)
12. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity to process information. Psychol. Rev. **63**(2), 81–97 (1956)
13. Nairne, J.S., Neath, I.: Sensory and working memory, volume 4, experimental psychology, chapter 15. In: Handbook of Psychology, 2nd edn., pp. 419–446 (2012)
14. Norman, D.A., Shallice, T.: Attention to action: willed and automatic control of behaviour. In: Consciousness and Self-Regulation. Advances in Research and Theory, vol. 4. Plenum Press (1986)
15. Ölveczky, P.C.: Real-time Maude and its applications. In: Escobar, S. (ed.) WRLA 2014. LNCS, vol. 8663, pp. 42–79. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12904-4_3
16. Ölveczky, P.C.: Designing Reliable Distributed Systems. UTCS. Springer, London (2017). https://doi.org/10.1007/978-1-4471-6687-0
17. Samsonovich, A.V.: Towards a unified catalog of implemented cognitive architectures. In: Biologically Inspired Cognitive Architectures (BICA 2010), pp. 195–244. IOS Press (2010)
18. Sun, R., Slusarz, P., Terry, C.: The interaction of the explicit and implicit in skill learning: A dual-process approach. Psychol. Rev. **112**, 159–192 (2005)
19. Verschure, P.: Distributed adaptive control: a theory of the mind, brain, body nexus. Biol. Inspired Cogn. Architect. **1**, 55–72 (2012)