Simulation and Analysis of Controlled Multi-Representational Reasoning Processes

Tibor Bosse[1], Catholijn M. Jonker[2], Jan Treur[1]

[1]*Vrije Universiteit Amsterdam, Department of Artificial Intelligence, De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands*

[2]*Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information, Montessorilaan 3, 6525 HR Nijmegen, the Netherlands*

Abstract

Multi-representational reasoning processes often show a variety of reasoning paths that can be followed. To analyse such reasoning processes with special attention for differences between individuals, it is required (1) to obtain an overview of the variety of different possibilities and (2) to address navigation and control within the reasoning process. This paper presents a simulation model and a formal analysis method for the dynamics of a controlled reasoning process in which multiple representations play a role. Reasoning Strategies to navigate through the space of possible reasoning states are modelled explicitly, and simulated. Simulation results are analysed by software tools on the basis of formalized dynamic properties. The variety of dynamic properties specified and the variety of traces simulated provides an overview for the individual differences between subjects that have been observed while solving multiplication problems.

1. Introduction

Human reasoning is often considered a process proceeding by accumulating a number of reasoning steps from start to end. An underlying assumption is that such a process can be analysed by studying each such step locally, in isolation from the rest of the reasoning process. Many reports of experimental research focus on one-trial-experiments where the number of reasoning steps is limited to one or, sometimes, at most two; e.g., (Rips, 1994; Johnson-Laird, 1983). However, a practical reasoning process often is not a straightforward accumulation of isolated steps. First, decisions to make a reasoning step may be not a local issue at the time point of the decision, but depend on the history and goals of the reasoning process as a whole. Second, often a multitude of reasoning paths is possible; only some of these actually reach the goal. Navigation and control in the sense of making a coherent set of choices at different time points to obtain one of the successful (and preferred according to one's own characteristics) paths is a nontrivial issue. Third, during the process steps may be taken that lead to a dead end, such that the reasoning process has to reconsider these steps, leading to revision of the reasoning path. These non-local aspects of a reasoning process require specific capabilities beyond, for example, the capability to locally apply modus ponens or modus tollens. Often some form of global reasoning planning and control is performed. Decisions to make or revise a specific reasoning step are made in the context of such a reasoning plan, which also has to be taken into account as part of a reasoning state.

In many cases the same information can be represented in different manners (e.g., in arithmetic, geometric or material form). Moreover, both internal (mental) and external (e.g., written or drawn) representations may play a role. The distinction between mental and external representations is also made in, e.g., (Hegarty, 2002). As the type of possible reasoning steps may be different for different forms of representation, these differences of representation have to be accounted for in different reasoning states. In such cases the number of possible reasoning states is not very small, and, as a consequence, the number of possible reasoning paths, may be quite large. Coherent controlled navigation involving non-local aspects of decisions for reasoning steps is of major importance to deal with such a large number of possibilities.

This paper reports analysis and simulation of controlled multi-representation reasoning processes, in which the issues put forward play an important role. An analysis method for the dynamics of reasoning is based on formal definitions of possible reasoning states and traces, and dynamic properties of these traces are specified in the Temporal Trace Language TTL (Jonker and Treur, 2002; Bosse, Jonker, Meij, Sharpanskykh, and Treur, J., 2006). This analysis method is supported by a software environment that is able to check traces against specified dynamic properties. For simulation the component-based agent design method DESIRE is used, cf. (Brazien et al., 2003). Traces generated by execution of a DESIRE model can be directly used as input of the analysis software environment.

In Section 2, the dynamic perspective on reasoning is discussed, with a focus on formalisation of the dynamics. Next, in Section 3, an example domain in reasoning with multiple representations is introduced. The example domain shows interaction between material, geometrical and arithmetical reasoning. It focuses on how to determine the outcome of multiplications such as 23 x 36, possibly using external arithmetic, geometric or material (based on Multi-base Arithmetic Blocks (MAB) material; e.g., Booker et al. 1997, English and Halford, 1995) representations. Section 4 is a brief introduction of the component-based agent modelling method DESIRE used for the simulation model. In Section 5, the design of the simulation model is presented. Various simulation traces have been generated, of which one example is briefly discussed. In Section 6, a number of dynamic properties for this type of reasoning are identified and formalised using TTL. Section 7 describes how these properties can be used to analyse existing (human or simulated) reasoning processes. In Section 8 some other component-based reasoning models are discussed. Finally, in Section 9 the approach is summarised and the contribution of the research presented in the paper is discussed.

2. Formalising Reasoning Dynamics

Analysis of the cognitive capability to perform reasoning has been addressed from different areas and angles. Within Cognitive Science, the two dominant streams are the syntactic approach (based on inference rules applied to syntactic expressions, as common in logic), e.g., (Rips, 1994), and the semantic approach (based on construction of mental models); e.g., (Johnson-Laird, 1983; Yang and Johnson-Laird, 1999).

Reasoning steps in natural contexts are usually not restricted to the application of logical inference rules. For example, a step in a reasoning process may involve translation of information from one representation form (e.g., geometrical) into another one (e.g., arithmetical). Or, an additional assumption can be made, thus using a dynamic set of premises within the reasoning process. Decisions made at specific points in time during the process, for example, on which representations to use or which assumptions to make, are an inherent part of the reasoning. Such reasoning processes or their outcomes cannot be understood, justified or explained without taking into account these dynamic aspects.

To formalise the dynamics of a reasoning process, traces are used. *Reasoning traces* are time-indexed sequences of *reasoning states* over a time frame; for stepwise reasoning processes the set of natural numbers as a time frame is an appropriate choice. The set of all possible reasoning states defines the space where the reasoning takes place. Reasoning traces can be viewed as trajectories in this space, for which every (reasoning) step from one reasoning state to the next one is based on an *allowed transition*. If the possible reasoning states and the allowed reasoning steps or transitions are characterised, the set of proper reasoning traces can be defined as the set of all possible sequences of reasoning states consisting only of allowed transitions.

2.1 Reasoning States

A *reasoning state* formalises an intermediate state of a reasoning process. The content of such a reasoning state usually can be analysed according to different aspects or dimensions. A reasoning state can include both internal (e.g., specific mental representations) and external elements (e.g., written or drawn notes). For example, part of the state may contain an external material representation, another part an external arithmetic representation, and yet another part an internal

geometric representation. Furthermore, as pointed out in the Introduction, also control information has to be taken into account in a reasoning state. Accordingly, the reasoning state is structured as a composition of (i.e., a tuple of) a number of parts, indexed by some set I. This index set includes different aspects or views taken on the state, e.g., I is the set

{control, extmaterial, extgeometric, extarithmetic, intmaterial, intgeometric, intarithmetic}.

The set of reasoning states RS can be characterised as a Cartesian product $RS = \Pi_{i \in I} RS_i$ where RS_i is the set of all states for the aspect indicated by i. For example, $RS_{extgeometric}$ may denote the set of all possible external (drawn) geometric representations. This Cartesian product formalises the multi-dimensional space where the reasoning takes place. For a reasoning state, which is a vector $S = (S_i)_{i \in I} \in RS$ in this space, the S_i are called its *parts*.

2.2 Reasoning Steps

A transition from one reasoning state to another reasoning state, i.e., an element $< S, S' >$ of $RS \times RS$, formalises one *reasoning step*; sometimes also denoted by $S \rightarrow S'$. Transitions differ in the set of parts that are involved. The most complex transitions change all parts of the state in one step. However, within stepwise reasoning processes, usually transitions only involve a limited number of parts of the state, e.g., one to three. In the current approach we concentrate on this class of transition types.

For example, when a modification in the reasoning state is made solely within an internal geometric representation, only the internal geometric part of the state changes (geometric reasoning step):

intgeometric $\rightarrow$ intgeometric

Other types of transitions involve more than one part. For example, if an external geometric representation is extended on the basis of an internal geometric representation, then two parts of the state are involved: the external geometric arithmetic part and the internal geometric part:

extgeometric x intgeometric $\rightarrow$ extgeometric

(e.g., the external geometric representation is extended or modified with results from the internal geometric representation)

If control information is incorporated in the modelling approach the number of involved parts is even higher, since every transition involves the control part; e.g.:

extgeometric x intgeometric x control $\rightarrow$ extgeometric

(e.g., the external geometric representation is extended or modified with results from the internal geometric representation and some control information)

2.3 Reasoning Traces

Reasoning dynamics results from successive reasoning steps, i.e., successive transitions from one reasoning state to another. Thus a *reasoning trace* is constructed: a time-indexed sequence of reasoning states $(\gamma_t)_{t \in T}$, where T is the time frame used (the natural numbers). A reasoning trace can be viewed as a trajectory in the multi-dimensional space $RS = \Pi_{i \in I} RS_i$ of reasoning states. An example of such a reasoning trace will be discussed in Section 3.2. Reasoning traces are sequences of reasoning states subject to the constraint that each pair of successive reasoning states in this trace forms an allowed transition. A trace formalises one specific line of reasoning.

3. Example Domain: Multiplication

In this section, an example domain in multi-representation reasoning is used to illustrate the approach put forward: how to determine the outcome of multiplications such as 23 x 36. When solving such multiplications, human may use multiple different representations in their reasoning, depending on the approach used during the education. This example focuses on the interaction between arithmetical, geometrical and material reasoning. Experiences on using such processes with children (8-9 years old) in class rooms have been reported, e.g., by Dekker et al. (1982), see also (Hutton, 1977). Also teaching quadratic equations can be supported by such visualisations as discussed, e.g., by Bruner (1968), pp. 59-63. For further explorations of the idea to use visualisations in pre-algebraic reasoning, see (Koedinger and Terao, 2002).

3.1 Basic Skills

For the example domain, a number of basic skills have been identified, that can be used within the reasoning. In terms reasoning steps (as discussed in the previous section), these basic skills consist of three types of one-component transitions of reasoning states, and four transition types involving two components:

- *arithmetical* reasoning steps: arithmetic → arithmetic
- *geometrical* reasoning steps: geometric → geometric
- *material* reasoning steps: material → material
- *translations* of an arithmetical representation into a geometrical representation: geometric x arithmetic → geometric
- *translations* of a geometrical representation into an arithmetical representation: arithmetic x geometric → arithmetic
- *translations* of an arithmetical representation into a material representation: material x arithmetic → material
- *translations* of a material representation into an arithmetical representation: arithmetic x material → arithmetic

The idea is that more experienced reasoners possess more basic skills than less experienced reasoning. Less experienced reasoners require only simple arithmetical steps. They can perform the more complicated steps via the geometrical or material representation. The skills can be defined (informally) in the form of the following transitions:

A. *Arithmetic skills* (arithmetic → arithmetic)

bs7. splitting a number in 'tens' and single digits: 23 = 20 + 3
bs8. translating a multiplication of two complex number to the multiplication of the two sums of a 'ten' and a single digit: 23x36 = (20+3)x(30+6)
bs9. multiplication of two numbers starting with a nonzero digit, followed by zero or more zeros, such as 20x8, 60x30.
bs10. applying the distribution law: (20+3)x(30+6) = (20x30)+(20x6)+(3x30)+(3x6)
bs11. extracting partial multiplication problems from a complex expression: (20x30)+(20x6)+(3x30)+(3x6) => (20x30)
bs12. filling in the solution to a partial multiplication problem in a complex expression
bs13. addition of a list of numbers of up to 4 digits, such as 600 + 120 + 90 + 18
bs14. concluding that the solution of the addition is the solution of the initial multiplication problem

B. *Geometric skills* (geometric → geometric)

bs4. partitioning a rectangle in non-overlapping areas based on partitionings of its sides

C. *Material skills* (material → material)

bs19. placing blocks inside the frame of a rectangle

D. *Translation skills* (geometric x arithmetic → geometric)

bs1. drawing a rectangle with arithmetically given dimensions
bs2. partitioning a line segment according to a splitting of its length
bs3. determining the surface of a rectangle from the multiplication of the lengths of its sides

E. *Translation skills* (arithmetic x geometric → arithmetic)

bs5. translating the area of a rectangle into the multiplication of the lengths of its sides
bs6. translating the area of a combination of nonoverlapping areas into the sum of the areas

F. *Translation skills* (material x arithmetic → material)

bs15. building the frame of a rectangle with arithmetically given dimensions
bs16. determining the surface of a group of identical blocks from the multiplication of the amount of blocks and the area of an individual block

G. *Translation skills* (arithmetic x material → arithmetic)

bs20. translating the area of a group of identical blocks into the multiplication of the amount of blocks and the area of an individual block
bs23. translating the area of a combination of groups of different block sizes into the sum of the areas of the groups

Notice that, in this notation no difference is made between the internal and the external elements of the reasoning states. However, the skills can easily be extended with this information. For example, basic skill bs1 can be extended in the two following ways:

bs1'. drawing a rectangle with arithmetically given dimensions on a piece of paper
(intgeometric x intarithmetic → extgeometric)
bs1". imagining a rectangle with arithmetically given dimensions
(intgeometric x intarithmetic → intgeometric)

A variety of (part of the) possible reasoning paths determined by these transitions is depicted in a simplified manner in Figure 1. For the sake of simplicity transitions between geometric and material representations have been left out. The numbers refer to basic skills. The boxes refer to (part of) the reasoning states. For example, the transition labelled "4" refers to skill bs4, i.e., partitioning a rectangle in non-overlapping areas, based on a partitioning of its sides, and the transitions labelled "7" refer to skill bs7, i.e., splitting a number in tens and digits.

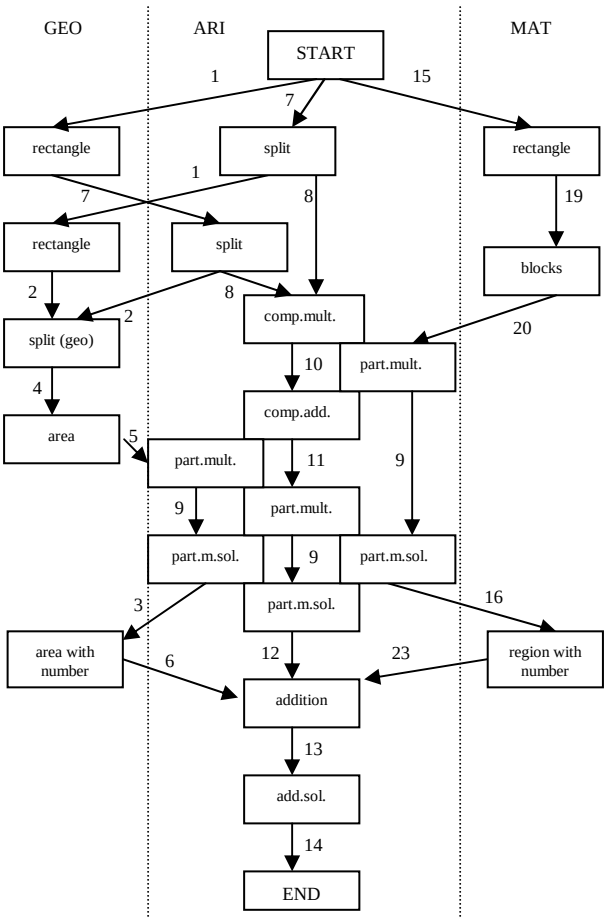

Figure 1 Variety of reasoning paths

3.2 Example Multi-Representational Reasoning Process

To illustrate the idea of the basic skills, Figure 2 presents a detailed reasoning trace. The starting problem for this trace was the following: "What is the outcome of the multiplication 23 x 36?" In this example, only geometrical and arithmetical representations are used. The example corresponds to a particular navigation path through Figure 1.

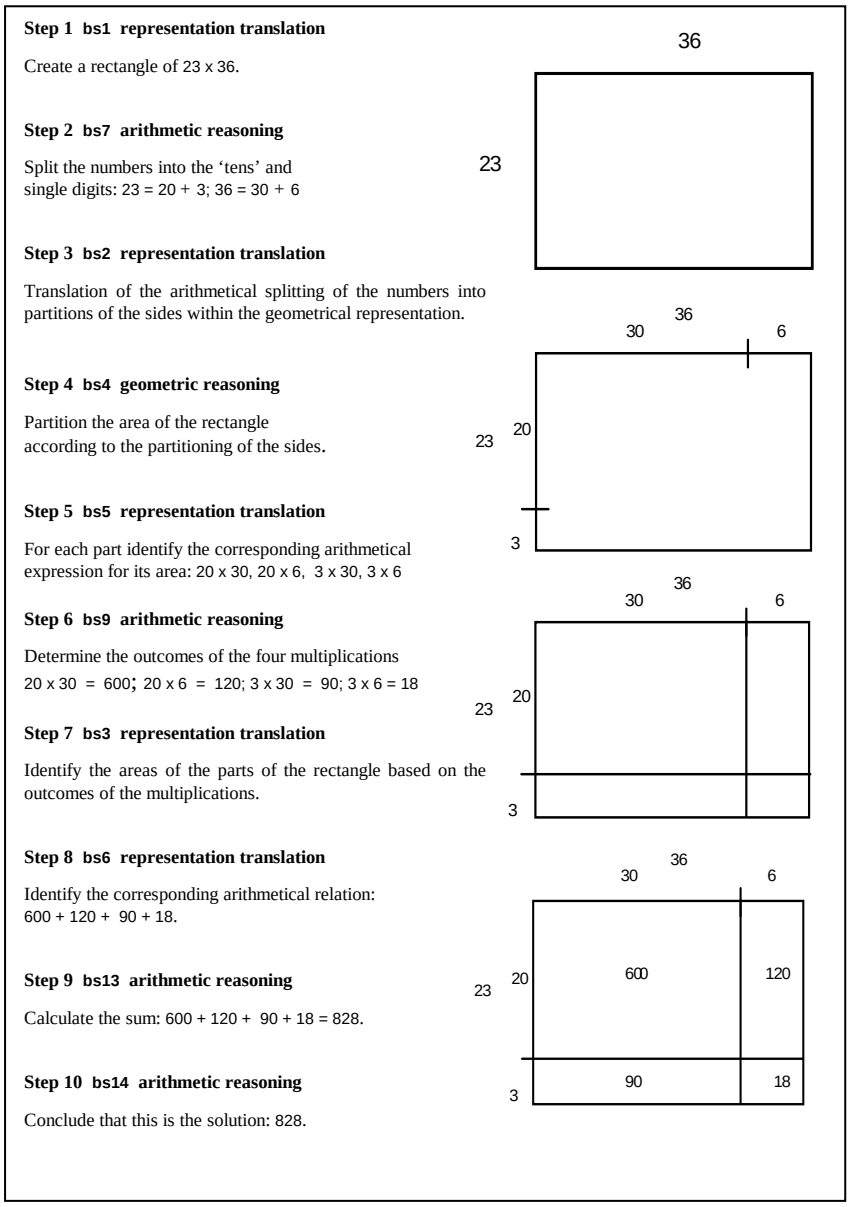

Figure 2 Example reasoning trace

4. Component-Based Design of Agents

The simulation of multirepresentation reasoning described in this paper has been developed using the component-based design environment DESIRE for agent systems (DEsign and Specification of Interacting REasoning components); for the underlying principles, see (Brazier, Jonker and Treur, 2000, 2002). DESIRE distinguishes itself by the extensive possibilities to specify internal agent models, so that also complex agents with sophisticated, knowledge-intensive reasoning capabilities and behaviour can be designed. It has been found useful in a number of applications of agent systems and reasoning systems within agents. Examples of such applications vary from BDI-agents (Brazier, Dunin-Keplicz, Treur, and Verbrugge, 1999) and normative agents (Castelfranchi, Dignum, Jonker, and Treur, 2000) to reasoning models for reasoning by assumption (Jonker and Treur, 2003) and nonmonotonic reasoning (Engelfriet and Treur, 2003). In this section DESIRE is briefly explained, taken from (Brazier, Jonker and Treur, 2000, 2002) where also more details can be found.

4.1 The DESIRE Modelling Approach

The development of an agent system is supported by graphical design tools within the DESIRE software environment. Translation to an operational system is straightforward; the software environment includes implementation generators with which formal specifications can be translated into executable code of a prototype system. In DESIRE, a design consists of knowledge of the following three types: process composition, knowledge composition, the relation between process composition and knowledge composition. These three types of knowledge are discussed in more detail below.

Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of (is composed of) lower level processes. Processes can be described at different levels of abstraction; for example, the process of the multi-agent system as a whole, processes defined by individual agents and the external world, and processes defined by task-related components of individual agents. The identified processes are modelled as *components*. For each process the *input and output information types* are modelled. The identified levels of process abstraction are modelled as *abstraction/specialisation relations* between components: components may be *composed* of other components or they may be *primitive*. Primitive components may be either reasoning components (i.e., based on a knowledge base), or, components capable of performing tasks such as calculation, information retrieval, optimisation. These levels of process abstraction provide process hiding at each level. The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *process composition*. This composition of processes is described by a specification of the possibilities for *information exchange* between processes (*static view* on the composition), and a specification of *task control knowledge* used to control processes and information exchange (*dynamic view* on the composition).

Knowledge composition identifies the knowledge structures at different levels of (knowledge) abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction levels may correspond to the process abstraction levels, but this is often not the case. The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. Knowledge structures can be identified and described at different levels of abstraction. At higher levels details can be hidden. An *information type* defines an ontology (lexicon, vocabulary) to describe objects or terms, their sorts, and the relations or functions that can be defined on these objects. Information types can logically be represented in order-sorted predicate logic. A *knowledge base* defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented by formulae in order-sorted predicate logic, which can be normalised by a standard transformation into rules. Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of more specific knowledge bases. The compositional structure is based on the different levels of knowledge abstraction distinguished, and results in information and knowledge hiding.

Each process in a process composition uses knowledge structures. Which knowledge structures are used for which processes is defined by the relation between process composition and knowledge composition.

Instead of designing each and every new agent application from scratch, an existing generic model can be used. Generic models can be distinguished for specific types of agents, of specific agent tasks and of specific types of multi-agent organisation. The use of a generic model in an application structures the design process: the acquisition of a conceptual model for the application is based on the generic structures in the model. A model can be generic in two senses:

- generic with respect to the *processes or tasks*
- generic with respect to the *knowledge structures*

Genericity with respect to processes or tasks refers to the level of process abstraction: a generic model abstracts from processes at lower levels. A more specific model with respect to processes is a model within which a number of more specific processes are distinguished, at a lower level of

process abstraction. This type of refinement is called *specialisation*. Genericity with respect to knowledge refers to levels of knowledge abstraction: a generic model abstracts from more specific knowledge structures. Refinement of a model with respect to the knowledge in specific domains of application, is refinement in which knowledge at a lower level of knowledge abstraction is explicitly included. This type of refinement is called *instantiation*.

Reuse as such, reduces the time, expertise and effort needed to design and maintain system designs. Which components, links and knowledge structures from the generic model are applicable in a given situation depends on the application. Whether a component can be used immediately, or whether instantiation, modification and/or specialisation is required, depends on the desired functionality. Other existing (generic) models can be used for specialisation of a model; existing knowledge structures (e.g., ontologies, thesauri) can be used for instantiation. Which models and structures are used depends on the problem description: existing models and structures are examined, rejected, modified, specialised and/or instantiated in the context of the problem at hand.

4.2 The Generic Agent Model GAM

The characteristics of weak agency, (Wooldridge, Jennings, 1995a,b), provide a means to reflect on the tasks an agent needs to be able to perform. Pro-activeness and autonomy are related to a self model, goals, and plans. Reactivity and social ability are related a world model, agent models, communication with other agents, and interaction with the external world. The design of the generic agent model (GAM) in a component-based approach entails consideration of the processes and knowledge an agent needs to perform and the composition of related components and knowledge structures.

Process composition within the generic agent model identifies the processes within an agent at the highest level of abstraction, and the manner in which they are composed to obtain the agent process (composition relation). The processes modelled within the generic agent model are depicted as components in Figure 3. The processes involved in controlling an agent (e.g., determining, monitoring and evaluating its own goals and plans) but also the processes of maintaining a self model are the task of the component own process control. The processes involved in managing communication with other agents are the task of the component agent interaction management. Maintaining knowledge of other agents' abilities and knowledge is the task of the component maintenance of agent information. Comparably, the processes involved in managing interaction with the external (material) world are the task of the component world interaction management. Maintaining knowledge of the external (material) world is the task of the component maintenance of world information. The specific task for which an agent is designed (for example: design, diagnosis, information retrieval), is modelled in the component agent specific task. Existing (generic) task models may be used to further structure this component. In addition, a component co-operation management may be distinguished for all tasks related to social processes such as co-operation in a project, or negotiation; e.g., (Brazier, Cornelissen, Jonker, and, Treur, 2000; Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak, and Treur, 2002)

The four characteristics of weak agency are related to these components in the following sense. Perception of the environment is performed by world interaction management (managing the perception process), maintenance of world information and maintenance of agent information (representation of perception information obtained from the environment). Actions in the world are managed by world interaction management. Social actions are managed by the tasks agent interaction management and cooperation management. The task cooperation management is not explained further in this chapter. Performing the agent's processes is initiated and co-ordinated by the task own process control; thus the agent's autonomous and pro-active behaviour is modelled.

A number of generic *information types* can be distinguished for the input and output of the generic agent model (based on external concepts) and for the generic processes within the agent (based on internal concepts). An agent capable of communication with other agents may receive incoming communication info and may send outgoing communication info. Moreover, the agent may observe and perform actions in the external (material) world. The information type observation info models the observations that are to be performed in the component external world. The information type observation result info models the incoming results of observations. The information type action info models the actions the

agent performs. In Table 1 an overview of the agent's interface information types is specified, based on the external primitive agent concepts.

The information types that express communication information are composed of information types on the subject of communication, and an information type to specify the agent from, or to whom, the communication is directed.

process	*input information types*	*output information types*
agent	incoming communication info observation result info	outgoing communication info observation info action info

Table 1 Specification of interface information types of the agent

The interface information types of the components within the agent are listed in Table 2. Within the agent component, the component own process control uses belief information on other agents and the external (material) world, as input. This information is modelled in the information type belief info which is composed of belief info on world and belief info on agents. The output of the component own process control includes the agent's characteristics (modelled in the information type own characteristics), used by the components agent interaction management and world interaction management. In addition to this information type, the component agent interaction management also receives the incoming communication received by the agent (and forwarded directly to the component agent interaction management), modelled in the input interface in the information type incoming communication info, and world and agent information,

process	*input information types*	*output information types*
own process control	belief info	own characteristics
agent interaction management	incoming communication info own characteristics belief info	outgoing communication info maintenance info
world interaction management	observation result info own characteristics belief info	observation info action info maintenance info
maintenance of agent information	agent info	agent info
maintenance of world information	world info	world info

Table 2 Specification of interface information types within the generic agent model

modelled in the input information type belief info. The output generated by the component agent interaction management includes the output for the agent as a whole (outgoing communication info), extended with maintenance info which is composed of maintenance info on agents and maintenance info on world (communicated information on the world and other agents that needs to be maintained).

The component maintenance of agent information receives new information on other agents (the agent's beliefs on other agents) in its input interface. These beliefs on other agents are made available to other components in the output interface of the component maintenance of agent information. Likewise the component world interaction management receives the agent's characteristics in the input information type own characteristics, observation results received by the agent (and forwarded directly to the component world interaction management) in the input interface type observation result info, and information the agent has about the world and agents in the information type belief info. The output generated by the component world interaction management includes the output for the agent as a whole (action info, observation info), extended with maintenance info (information obtained from observation of the world and other agents that needs to be maintained). The component maintenance of world information receives new information on the world (the agent's beliefs on the world) in its input interface. Beliefs on the world are available in the output interface of the component maintenance of world information. More detaild on information types within the generic agent model GAM can be found in (Brazier, Jonker and Treur, 2000).

Information exchange within the agent is specified by the information links depicted as arrows in Figure 3. Observation results are transferred through the information link observation result info to wim from the agent's input interface to the component world interaction management. In addition, this component receives belief information from the component maintenance of world information through the information link world info to wim, and the agent's characteristics from the component own process control through the link own process info to wim. The selected actions and observations (if any) are transferred to the output interface of the agent through the information link observations and actions.

The component maintenance of world information receives (meta-)information on observed world information from the component world interaction management, through the information link observed world info and meta-information on communicated world information (through the link communicated world info) from the component agent interaction management. Epistemic information from maintenance of world information, epistemic world info, is transferred to input belief info on world of the components world interaction management, agent interaction management and own process control, through the information links world info to wim, world info to aim and world info to opc.

Comparably the component maintenance of agent information receives meta-information on communicated information from the component agent interaction management, through the information link communicated agent info and meta-information on observed agent information (through the link observed agent info) from the component world interaction management. Epistemic information, epistemic agent info, is output of the component maintenance of agent information, becomes input belief info on agents of the components world interaction management, agent interaction management and own process control, through the information links agent info to wim, agent info to aim and agent info to opc.

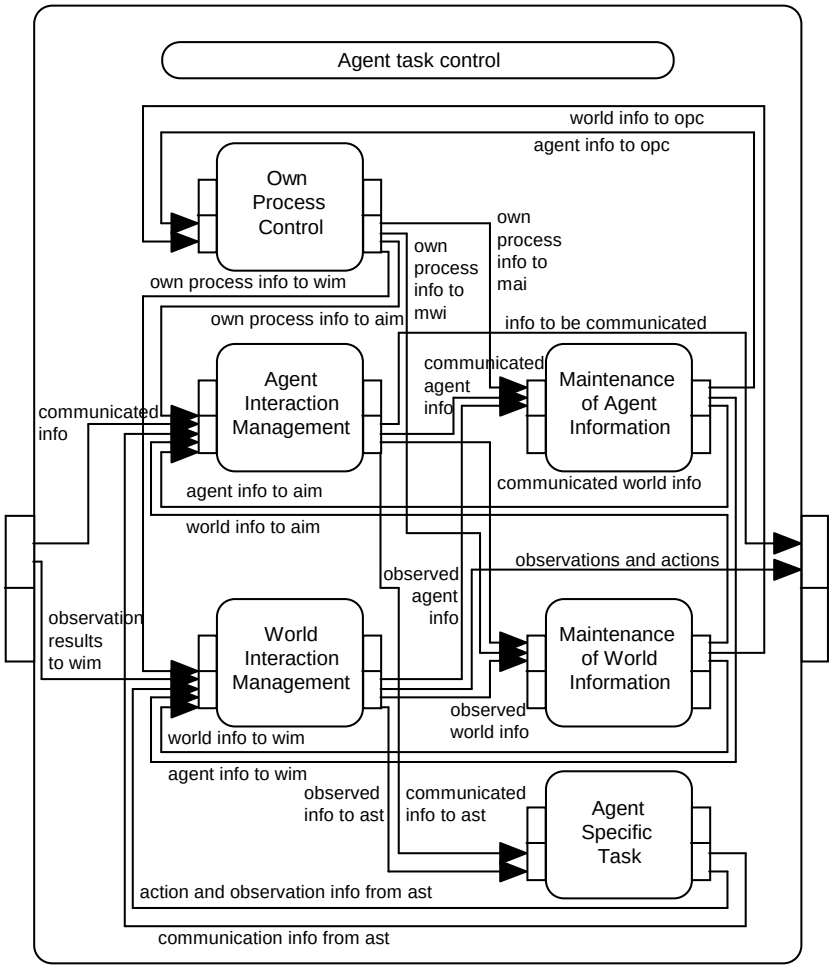

Figure 3 Information exchange at the highest process abstraction level within the agent

4.3 Comparison to Some Other Agent Modelling Approaches

The Concurrent MetateM framework is another modelling framework for multi-agent systems, based on modal temporal logic; cf. (Fisher, 1995, 2005; Barringer, Fisher, Gabbay, Owens, and Reynolds, 1996; Galton, 2003, 2006). A comparison with DESIRE is discussed for the structure of agents, inter-agent communication and meta-level reasoning. Morerover, a comparison to object-oriented approaches is discussed. For more extensive comparisons of DESIRE to other approaches, see (Mulder, Treur, and Fisher, 1998; Shehory and Sturm, 2001).

For the *structure of agents*, in DESIRE, the knowledge structures that are used in the knowledge bases and for the input and output interfaces of components are defined in terms of information types, in which sort hierarchies can be defined. Signatures define sets of ground atoms. An assignment of truth values *true*, *false* or *unknown* to atoms is called an information state. Every primitive component has an internal information state, and all input and output interfaces have information states. Information states evolve over time. Atoms are persistent in the sense that any ground atom in a certain information state is assigned to the same truth value as in the previous information state, unless its truth value has changed because of updating an information link. Concurrent MetateM does not have information types, there is no predefined set of ground atoms and there are no sorts. The input and output interface of an object consists only of the names of predicates. Two-valued logic is used with a closed world assumption, thus an information state is defined by the set of ground atoms that are true.

In a DESIRE specification of a multi-agent system, the agents are (usually) subcomponents of the top-level component that represents the whole (multi-agent) system, together with one or more components that represent the rest of the environment. A component that represents an agent can be a composed component: an agent process hierarchy is mapped into a hierarchy of components. In a Concurrent MetateM model, agents are modelled as objects that have no further structure: all its tasks are modelled with one set of rules.

The *communication* between agents in DESIRE is defined by the information links between them: communication is based on point-to-point or broadcast message passing. Communication between agents in Concurrent MetateM is done by broadcast message passing. When an object sends a message, it can be received y all other objects. On top of this, both multi-cast and point-to-point message passing can be defined.

The compositional approach to agent design in this paper has some aspects in common with object oriented design methods; e.g., (Booch, 1994). However, there are differences as well. Examples of approaches to object-oriented agent specifications can be found in, e.g., (Aridor and Lange, 1998). A first interesting point of discussion is to what the difference is between agents and objects. Some tend to classify agents as different from objects. For example, (Jennings and Wooldridge, 1998) compare objects with agents on the dimension of autonomy in the following way:

> 'An object encapsulates some state, and has some control over this state in that it can only be accessed or modified via the methods that the object provides. Agents encapsulate state in just the same way. However, we also think of agents as encapsulating behaviour, in addition to state. An object does not encapsulate *behaviour*: it has no control over the execution of methods – if an object x invokes a method m on an object y, then y has no control over whether m is executed or not – it just *is*. In this sense, object y is not autonomous, as it has no control over its own actions. In contrast, we think of an agent as having *exactly* this kind of control over what actions it performs. Because of this distinction, we do not think of agents as invoking methods (actions) on agents – rather, we tend to think of them *requesting* actions to be performed. The decision about whether to act upon the request lies with the recipient.'.

Some others consider agents as a specific type of objects that are able to decide by themselves whether or not they execute a method (objects that can say 'no'), and that can initiate action (objects that can say 'go').

A difference between the component-based design method DESIRE and object-oriented design methods in representation of basic functionality is that within DESIRE declarative, knowledge-based specification forms are used, whereas method specifications (which usually have a more procedural style of specification) are used in object-oriented design. Another difference is that within DESIRE the composition relation is defined in a more specific manner: the static aspects by information links, and the dynamic aspects by (temporal) task control knowledge, according to a pre-specified format. A similarity is the (re)use of generic structures: generic models in DESIRE, and patterns; cf. (Gamma, Helm, Johnson, and Vlissides, 1994) in object-oriented design methods, although their functionality and compositionality are specified in different manners, as discussed above.

5. Simulation Model

The simulation model[1] for multi-representational reasoning is based on the component-based agent modelling approach DESIRE briefly introduced in Section 4; cf. (Brazier et al., 2000, 2002). At the highest level of abstraction, two components play a role in the system, i.e., the reasoning agent (called *Alan*) and the *External World*. Figure 3 depicts an overview of the components of the simulation model.

Alan can perform actions and observations, executed in the external world, and receive observation results as input from the external world. After Alan generates a certain action to be performed (e.g., draw a rectangle with sides 23 x 36), this action is transferred to the external world and executed there. The result of the action (e.g., a rectangle with corners A, B, C, D and sides 23 x 36 drawn on a piece of paper) will occur, with a certain delay, within the external world. Thus, the execution of physical actions by the agent is modelled as part of the component external world. Several kinds of physical actions are involved: writing things down (e.g., numbers), drawing pictures, and placing objects (e.g., blocks). Besides performing actions, Alan can pro-actively observe the world. The agent does this by explicitly determining what aspects of the world it is interested in: its observation focus. This focus is then transferred to the external world, which in return provides the corresponding observation result.

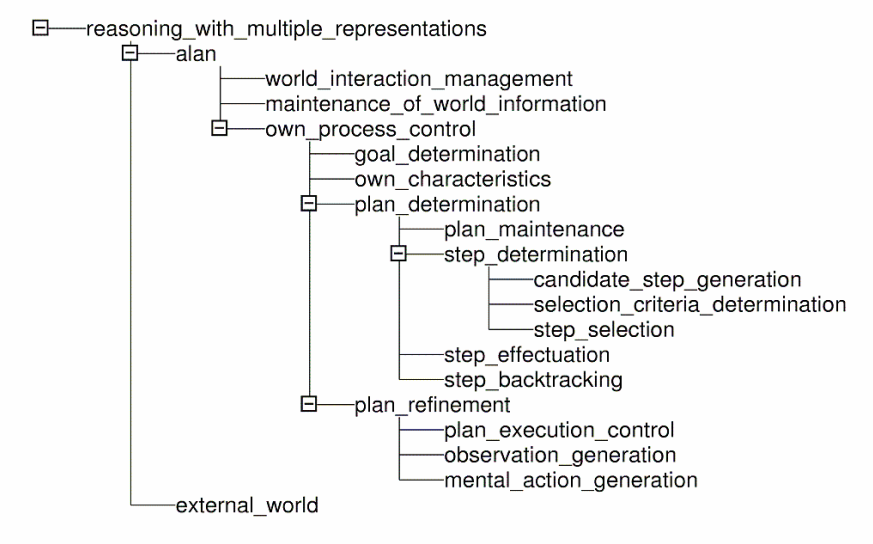

Figure 4 Overview of the components of the simulation model

5.1 Reasoning Agent

The approach used in this paper assumes that for every action a mental and a physical part can be distinguished and modelled (e.g., imagining a rectangle with sides 23 x 36 vs. actually drawing such a

[1] A complete specification of the model (with clickable components) can be found at: www.cs.vu.nl/~wai/GTM/rmr/

rectangle). Whilst the external world is concerned with the physical parts of the actions, everything that is represented within the agent is mental. Moreover, the different representations used in the reasoning process (i.e., arithmetic, geometric, material) should be modelled differently. To be able to make a clear distinction between these different concepts, different ontologies are used, e.g., rectangle(A, B, C, D, 23, 36) denotes a specific rectangle in the world (i.e., using a geometric representation), whereas entity(shape([]), parameters(23, 36)) denotes the internal representation of such as rectangle. Likewise, multiplication(23, 36) represents a specific multiplication problem in the world (i.e., using an arithmetic representation), whereas entity(shape("X*Y"), parameters(23, 36)) denotes the internal representation of such a multiplication. Internal representations can be created on the basis of an observation, but also on the basis of internal reasoning.

The composition of the reasoning agent Alan is based on the generic agent model as described in (Brazier et al., 2000) and briefly introduced in Section 4.2 above. Three of the generic agent components are used in our model, namely *World Interaction Management*, *Maintenance of World Information* and *Own Process Control*. The other generic agent components were not needed within this model.

The task of the component *Maintenance of World Information* is to maintain a (partial) world model, i.e., a snapshot of the present world state. In this domain, this world model is restricted to the observed information about objects that the agent has manipulated itself, such as the numbers it has written down. Moreover, since the agent does not necessarily have to perform each intermediate step physically, some imaginary world model must be maintained as well. This model describes the world like it would be after the physical execution of some steps, without these steps actually being performed. As both models contain information about a (possible) state of the world, both are maintained by Maintenance of World Information. An important issue is the amount of time that the world models persist within the component. In the current model, this duration is very short; thus, the component can be compared with part of the short-term memory: the information enters, is (possibly) used by another component, and very quickly disappears. Hence, whenever the information is needed later on, it has to be created again (either by observation or by imagination). This loss of information is modelled by clearing the contents of the component soon after it has entered. However, the duration of this period can easily be modified.

According to the generic agent model, tasks of the component *Own Process Control* are the processes the agent uses to control its own activities (e.g., determining, monitoring and evaluating its own goals and plans), but also the processes of maintaining a self model. The way the tasks are performed is described in detail in the next section.

5.2 Own Process Control

Own Process Control consists of four sub-components: *Goal Determination*, *Own Characteristics*, *Plan Determination* and *Plan Refinement*, see Figure 4. These components are responsible for, respectively, determining the agent's goals, its own characteristics, planning the reasoning process at an abstract level, and actually performing the reasoning process. Their details of how they work are described in this section.

5.2.1 Goal Determination

For the application in question, *Goal Determination* is a relatively simple component. It contains information about the initial multiplication problem the agent desires to solve. The fact that the initial problem is represented here reflects the situation that the desire to solve this particular problem has popped up within the agent's mind spontaneously. However, in many cases the determination of goals is a more complex process. Therefore, the component can easily be extended to simulate a more dynamic form of goal determination (e.g., involving the possibility to modify and drop goals).

5.2.2 Own Characteristics

The component *Own Characteristics* contains a self-model, which includes several aspects. In the first place, it includes (self-)information on the basic skills that the reasoning agent thinks to possess.

Note that this does not necessarily mean that the agent indeed has all these skills. For instance, it is well possible that the agent believes to be able to apply the distribution law of arithmetic, whilst during execution it turns out that it does not (i.e., the agent overestimated itself). Also, the opposite is possible. In that case, an agent possesses certain skills of which it does not know it has them. As a consequence, it will never use these skills. In the case of the over confident agent, when a certain skill has failed (i.e., the agent planned to use it, but at the end, it could not), Own Characteristics revises the self-knowledge of the agent by asserting that it does not have the skill after all. Second, Own Characteristics is used to store the agent's profile with respect to its problem solving strategy for the multiplication problem. Two aspects are represented: (1) a list of priorities among the different representations that can be used while solving the problem (e.g., the profile ari-geo-mat indicates that the agent prefers arithmetical representations to geometrical and material ones), and (2) to what extent steps in the reasoning process have to be performed physically. This way, several types of agents can be modelled, varying from those that write down every step to those that write down nothing. As a final remark, notice that, although DESIRE offers the opportunity to dynamically add changes in the specification (and thereby realise an open state space), this has not been done within the current model.

5.2.3 Plan Determination

Before actually solving the problem, the reasoning agent makes an abstract plan (e.g., a particular navigation route through Figure 1). *Plan Determination* is responsible for this planning process. Its input consists of the agent's own goal and characteristics. Based on this information, and knowledge about pre- and postconditions of the basic skills, Plan Determination explores the entire reasoning process at an abstract level. It uses abstract knowledge about when a certain basic skill can be applied (preconditions), and what the effect of this application will be (postconditions). The pre- and postconditions are expressed in an abstract way; e.g., they do not contain any numbers. While planning, Plan Determination continuously matches the current state of the explored plan against the preconditions of all basic skills, in order to determine which skills are applicable. It then uses its strategy profile as control knowledge, in order to select one of the applicable skills. Subsequently, the skill is evaluated by adding its (abstract) postcondition to the current state of the explored plan. This way, the component constructs a complete list of steps to be performed, that would solve the multiplication problem. Furthermore, the component uses backtracking in situations where no more basic skills are applicable. Finally, if no solution can be found at all, this is also indicated. The sub-components of Plan Determination will be described in Section 5.3.

5.2.4 Plan Refinement

Abstract plans, generated by Plan Determination, are transferred to the component *Plan Refinement*. This component, which consists of the sub-components *Plan Execution Control*, *Precondition Acquisition Initiation* and *Mental Action Execution*, is responsible for the refinement of the basic steps, i.e., it determines the specific mental and physical actions associated to a basic step of the abstract plan (e.g., it refines bs4 to bs4m). Moreover, it executes the detailed mental actions associated to the basic steps. This is done by repeating the following activities. First, Plan Execution Control selects the first step of the (remaining) plan to be executed. Second, Precondition Acquisition Initiation determines what observations have to be made to provide the agent with the necessary information for the application of the selected step. For instance, if the selected step is to draw a rectangle, it is important to know the dimensions of the rectangle. Third, as soon as this information has been obtained, Mental Action Execution creates a mental image of the result of the application of the mental action (with instantiated variables, e.g., 'a rectangle with sides 23 x 36', denoted by entity(shape([]), parameters(23, 36))). This mental image is then stored within Maintenance of World Information. Within this step, if necessary, ontology mapping is performed. For example, in case an action is executed in which a translation needs to be made from one representation to another (see the *Translation Skills* in Section 3.1), then this component makes sure that the result of the action is represented using the correct ontology. After that, Plan Execution Control decides whether to

perform the associated physical action as well, depending on the agent's own characteristics. Then, the physical action either is or is not executed (within the External World component), after which the next step of the plan is treated by Plan Execution Control. Finally, if the agent is unable to perform an action that it had planned to do because it lacks either the mental or the physical skill for that action, notification with the name of the skill that failed is transferred to Own Characteristics. As a consequence, this latter component will revise its self-model, so that Plan Determination can construct a new plan more adequately.

5.3 Plan Determination

Plan Determination consists of the components *Plan Maintenance*, *Step Determination*, *Step Effectuation* and *Step Backtracking*. Plan Maintenance keeps track of all kinds of information concerning the 'current' state of the explored reasoning process, such as the (abstract) steps that have been applied successfully, those that have failed and those that have not been applied yet. Step Determination determines the next step to be added to the current plan in three phases. First, it determines which steps are currently applicable, by matching the preconditions of abstract steps against the current state of the exploration. Second, based on the applicable steps and the agent's strategy profile, it decides whether it will make an arithmetic, geometric, or material step. And third, based on the chosen representation, it will select one single step. The components responsible for the three phases are called, respectively, *Candidate Step Generation*, *Selection Criteria Determination*, and *Step Selection*. Finally, the selected step is passed to Step Effectuation. However, if, independently of the representation, no steps are applicable, this failure is indicated, so that the backtracking component can become active. Step Effectuation explores the execution of the selected abstract step by adding the postcondition of the step to the current state of the simulation. Step Backtracking becomes active whenever no more steps are applicable and uses a standard backtracking algorithm.

5.4 Example Simulation Trace

Using the model described above, a number of simulations have been performed. An example of a resulting simulation trace is shown in Table 1. In this trace, both geometric and arithmetic skills are used to solve the problem, although there is a preference for the geometric skills. More simulation traces are included in Appendix A.

Table 1
Example simulation trace

Step	Information Derived	
0	strategy profile:	geo-ari-mat
0	available abstract skills:	all skills
0	available mental skills:	all skills
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
2	plan([bs1, bs7, bs2, bs4, bs5, bs9, bs3, bs6, bs13, bs14])	
3	is_represented_in_world(arithmetic, multiplication(23, 36))	
4	mental_representation(geometric, entity(shape("[]"), parameters(23, 36)))	
5	is_represented_in_world(geometric, rectangle('A', 'B', 'C', 'D', 23, 36))	
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(36, 30, 6)))	
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(23, 20, 3)))	
7	is_represented_in_world(arithmetic, split(36, 30, 6))	
7	is_represented_in_world(arithmetic, split(23, 20, 3))	
8	mental_representation(geometric, entity(shape("-"), name('A', 'B'), parameters(20, 3)))	
8	mental_representation(geometric, entity(shape("-"), name('A', 'D'), parameters(30, 6)))	
9	is_represented_in_world(geometric, split('A', 'B', 20, 3))	
9	is_represented_in_world(geometric, split('A', 'D', 30, 6))	
10	mental_representation(geometric, entity(shape("[]"), name('A11'), parameters(20, 30)))	
10	mental_representation(geometric, entity(shape("[]"), name('A12'), parameters(20, 6)))	
10	mental_representation(geometric, entity(shape("[]"), name('A21'), parameters(3, 30)))	
10	mental_representation(geometric, entity(shape("[]"), name('A22'), parameters(3, 6)))	
11	is_represented_in_world(geometric, area('A11', 20, 30))	
11	is_represented_in_world(geometric, area('A12', 20, 6))	
11	is_represented_in_world(geometric, area('A21', 3, 30))	
11	is_represented_in_world(geometric, area('A22', 3, 6))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 6)))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 30)))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 6)))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 30)))	
13	is_represented_in_world(arithmetic, partial_multiplication('A11', 20, 30))	
13	is_represented_in_world(arithmetic, partial_multiplication('A12', 20, 6))	
13	is_represented_in_world(arithmetic, partial_multiplication('A21', 3, 30))	
13	is_represented_in_world(arithmetic, partial_multiplication('A22', 3, 6))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 6, 18)))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 30, 90)))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 6, 120)))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 30, 600)))	
15	is_represented_in_world(arithmetic, multiplication_solution(3, 6, 18))	
15	is_represented_in_world(arithmetic, multiplication_solution(3, 30, 90))	
15	is_represented_in_world(arithmetic, multiplication_solution(20, 6, 120))	
15	is_represented_in_world(arithmetic, multiplication_solution(20, 30, 600))	
16	mental_representation(geometric, entity(shape("[]"), name('A11'), area_with_number(600)))	
16	mental_representation(geometric, entity(shape("[]"), name('A12'), area_with_number(120)))	
16	mental_representation(geometric, entity(shape("[]"), name('A21'), area_with_number(90)))	
16	mental_representation(geometric, entity(shape("[]"), name('A22'), area_with_number(18)))	
17	is_represented_in_world(geometric, area_with_number('A11', 600))	
17	is_represented_in_world(geometric, area_with_number('A12', 120))	
17	is_represented_in_world(geometric, area_with_number('A21', 90))	
17	is_represented_in_world(geometric, area_with_number('A22', 18))	
18	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))	
19	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))	
20	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))	
21	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))	
22	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))	
23	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))	

As can be seen in the table, the trace first contains a description of the characteristics of the agent (step 0), then the arithmetic problem is mentally represented (step 1) and the abstract plan is produced (step 2). Due to the strategy profile of the agent, the plan shows as many basic geometric skills as possible (this corresponds to a route through the left part of Figure 1). Every step is represented both mentally and physically, corresponding to the agent's characteristics. Since the agent has all skills both in abstracto and in concreto, no backtracking was necessary either during plan determination or plan execution.

Note that, although a large variety of reasoning paths is possible through the space of different reasoning states (see also Figure 1), during simulation, specific control knowledge is used to guide the reasoning trace in a deterministic manner. This control knowledge, which is used within the component Step Determination (see Section 5.3), basically consists of the agent's strategy profile, which states, for example, whether the agent prefers geometric representations over arithmetic representations. As a result, only one trace is generated per combination of initial parameter settings.

6. Dynamic Properties

To specify properties on the dynamics of a reasoning process, the temporal trace language TTL used by Jonker and Treur (2002) is adopted. This is a language in the family of languages to which also situation calculus (Reiter, 2001), event calculus (Kowalski and Sergot, 1986), and fluent calculus (Hölldobler and Tielscher, 1990) belong; see also (Galton, 2003, 2006) for more background in temporal modelling languages. In short, in TTL it is possible to express that in a given trace at a certain point in time the reasoning state has a certain (state) property. Moreover, it is possible to relate such state properties at different points in time. More details about TTL can be found in (Jonker and Treur, 2002; Bosse, Jonker, Meij, Sharpanskykh, Treur, 2006); for the process from informal to formal TTL-specifications, see also (Herlea Damian, Jonker, Treur, and Wijngaards, 2005).

As an example, the following (global) property of a reasoning trace γ is considered, which expresses that all multiplication problems in two digits eventually will be solved.

GP1 (successfulness)

at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point $t' \geq t$ exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z of this multiplication problem with $z = x*y$ is included.

The formalisation of this property in TTL is as follows.

$\forall t\ \forall x, y < 100$ state(γ, t, arithmetic) |== multiplication_problem(x, y)
$\Rightarrow\ \exists t' \geq t\ \exists z\quad z = x*y\ \ \&\ \$ state(γ, t', arithmetic) |== is_solution_for_multiplication_of(z, x, y)

Note that for simplicity no maximal allowed response time has been specified. If desired, this can be simply added by putting a condition $t' \leq r$ in the consequent with r the maximal response time. Similarly, other variants of overall properties can be specified, for example expressing that within the trace all multiplication problems will be solved without using any geometric or material representations. Moreover, instead of the arithmetical part of the reasoning state (arithmetic), again the specific internal or external arithmetical part (intarithmetic or extarithmetic) can be used, for example when expressing that only internal arithmetical representations are used. In the remaining of this paper, only the informal notation will be used for the properties.

6.1 Milestone Properties

Within the overall reasoning process a number of milestones can be defined, and properties can be identified that express whether the process from one milestone to another one has been performed properly. With respect to the geometrical reasoning, two intermediate milestones were defined: a reasoning state in which the problem has been represented in a geometric representation and it has been decomposed geometrically (after step 4 in the example trace), and a reasoning state in which a geometric representation with numbers in the areas occurs, i.e., in which the subproblems have been solved (after step 7 in the example trace). Accordingly, the following milestone properties have been formulated.

MP1

at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point $t' \geq t$ exists such that in the reasoning state in γ at t' a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD, with |AB| = x and |AD| = y
and this rectangle is partitioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC with $|AP| = x_1$, $|PB| = x_2$, $|AQ| = y_1$, and $|QD| = y_2$, where x_1, y_1 is the 10-part of x, resp. y, and x_2, y_2 is the digit part of x, resp. y.

Here, |AB| is the length of AB, and // is 'in parallel with'.

MP2

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD, with |AB| = x and |AD| = y,
and this rectangle is partitioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC with |AP| = x_1, |PB| = x_2, |AQ| = y_1, and |QD| = y_2, where x_1, y_1 is the 10-part of x, resp. y, and x_2, y_2 is the digit part of x, resp. y,
then a time point t'≥t exists such that in the reasoning state in γ at t' in each of these areas Aij a number zij is represented which equals $x_i * y_j$.

MP3

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with |AB| = x and |AD| = y
and this rectangle is partitioned into four nonoverlapping rectangle areas A_{11}, A_{12}, A_{21}, A_{22},
and in each of these areas A_{ij} a number zij is represented which equals $x_i * y_j$, where $x = x_1 + x_2$, and $y = y_1 + y_2$,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z with z = x∗y of the multiplication problem (x, y) is included.

6.2 Local Properties

In this section a number of properties are identified that characterise the reasoning in a more local manner: each property characterises one reasoning step. For the sake of simplicity, for the example reasoning process persistence of representations in reasoning states over time is assumed, so that persistence does not need to be formulated within each of the properties.

LP1 (arithmetic-geometric)

at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point t'≥t exists such that in the reasoning state in γ at t' a geometric representation of a rectangle ABCD with |AB| = x and |AD| = y is included.

This dynamic property expresses that in reasoning trace γ, if an arithmetically represented multiplication problem occurs, this eventually is translated into a geometric representation. The formalisation of this property in TTL is as follows.

∀t ∀x, y < 100 state(γ, t, arithmetic) |== multiplication_problem(x, y)
⇒ ∃t'≥t ∃A, B, C, D
state(γ, t', geometric) |== rectangle(A, B, C, D) & |AB| = x & |AD| = y

Further local properties are the following (not in any particular order).

LP2 (arithmetic-arithmetic)

at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a splitting of the numbers x and y in 'tens' and digits occurs, i.e., $x = x_1 + x_2$, $y = y_1 + y_2$ with x_1, y_1 multiples of 10 and x_2, y_2 <10.

LP3 (arithmetic-arithmetic)

at any point in time t
if the reasoning state in trace γ at t contains an arithmetic representation of a multiplication problem for (x, y), with x, y multiple of 10 or less than 10,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z with z = x∗y for this multiplication problem for (x, y) is included.

LP4 (arithmetic-arithmetic)

at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of an addition problem for a finite list z_1 ,..., z_n of numbers of up to 4 digits is included,
then a time point t'≥t exists such that in the reasoning state in γ at t' a solution $z = \Sigma_{1 \leq i \leq n} z_i$ of the addition problem is included.

LP5 (arithmetic-geometric)

at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a splitting of the numbers x and y occurs, i.e., $x = x_1 + x_2$, $y = y_1 + y_2$,
then a time point t'≥t exists such that in the reasoning state in γ at t' a geometric representation of a rectangle ABCD with |AB| = x and |AD| = y is included with points P on AB and Q on AD such that $|AP| = x_1$, $|PB| = x_2$, $|AQ| = y_1$, and $|QD| = y_2$.

LP6 (geometric-geometric)

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD,
then a time point t'≥t exists such that in the reasoning state in γ at t' the rectangle ABCD is partitioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC.

LP7 (geometric-arithmetic)

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD with |AB| = x and |AD| = y is included with points P on AB and Q on AD such that $|AP| = x_1$, $|PB| = x_2$, $|AQ| = y_1$, and $|QD| = y_2$,
and this rectangle is partioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC,
then a time point t'≥t exists such that in the reasoning state in γ at t' arithmetic representations of multiplication problems for (x_1, y_1), (x_1, y_2), (x_2, y_1), and (x_2, y_2) are included.

LP8 (geometric&arithmetic-geometric)

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD,
and this rectangle is partioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC,
and arithmetic representations of solutions z_{11}, z_{12}, z_{21}, z_{22} for the multiplication problems for (|AP|, |AQ|), (|AP|, |QD|), (|PB|, |AQ|), and (|PB|, |QD|) are included.
then a time point t'≥t exists such that in the reasoning state in γ at t' within the geometric representation in each area A_{ij}, the number z_{ij} is represented.

LP9 (geometric-arithmetic)

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included which is partioned into a number of areas $A_1, \ldots, A_n$,
and within each of these areas A_i a number z_i is represented,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of an addition problem for $z_1, \ldots, z_n$ is included.

LP10 (geometric& arithmetic-arithmetic)

at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with |AB| = x and |AD| = y that is partitioned into a number of nonoverlapping areas $A_1, \ldots, A_n$,
and within each of these areas A_i the number zi is represented,
and an arithmetic representation of a solution z of the addition problem for $z_1, \ldots, z_n$ is included,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z with $z = x*y$ of the multiplication problem (x, y) is included.

7. Dynamics Analysis

In this section it is described how the dynamic properties can be used for the analysis of existing reasoning processes.

7.1 Logical Relationships

A number of logical relationships have been established between the properties above. First of all, the three milestone properties together imply the global property:

MP1 & MP2 & MP3 ⇒ GP1 (0)

Next, each of these milestone properties is implied by a number of local properties:

LP1 & LP2 & LP5 & LP6 ⇒ MP1 (1)
LP3 & LP7 & LP8 ⇒ MP2 (2)
LP4 & LP9 & LP10 ⇒ MP3 (3)

These logical relationships can be depicted as an AND-tree, see Figure 5.

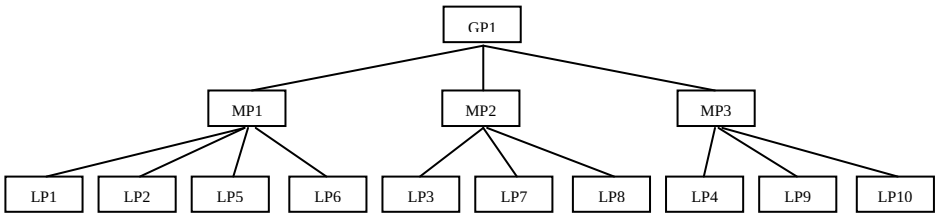

Figure 5 Logical relationships between dynamic properties

Identification of such logical relationships can be helpful in the analysis of errors within a given reasoning trace. For example, in case of a non-satisfactory reasoning trace it can first be checked whether **GP1** holds. If this global property does not hold, the three properties **MP1**, **MP2**, **MP3** can be checked. Given the logical relationship (0), at least one of them will be found not to hold. This pinpoints the cause of the error in part of the process, say **MP3**. Next, (only) the local properties relating to **MP3** are checked, i.e, **LP4**, **LP7**, **LP10**, **LP11**. Again, due to (3) one of them will be found not to hold, which localises the cause of the error. Notice that this diagnostic process is economic in the sense that the whole subtrees under **MP1** and **MP2** are not examined as long as there is no reason for that.

7.2 Dynamic Analysis Method

Based on the idea of logical relationships between properties, a general analysis method for the dynamics of reasoning processes can be formulated. This analysis method comprises the following steps:

1. Identify the different dimensions or *components* of reasoning states.
2. Determine the different *types of transitions*.
3. Identify relevant *dynamic properties* for the reasoning
   a. for the process as a whole (global properties)
   b. for milestones within the process
   c. for reasoning steps (local properties)
4. Determine logical *relationships* between the different dynamic properties, in an AND-tree form; e.g.,
   a. local properties imply a milestone property, and
   b. milestone properties imply a global property.
5. For a given reasoning trace, *check* which of the dynamic properties hold and which do not hold. This can take the form of a diagnosis following the tree structure of the relationships between the dynamic properties. A software environment is available to support this checking process.

For the case at hand, more than 70 dynamic properties have been specified, varying from global properties for the overall reasoning process to more local properties. The idea is that some of these properties are of a general nature (i.e., they can be used to assess whether a trace qualifies as a proper reasoning trace), whereas the other properties are used to characterise the different types of possible traces (i.e., they are used to identify individual differences). A large number of automated checks have been performed, thereby checking dynamic properties as described in Section 6 against simulated traces as shown in Section 5.4, to reveal which properties hold for which traces. The results

were in line with our expectations; for example, in the traces where all basic skills are present (e.g. the trace in Table 1), all properties of a general nature (such as the successfulness property GP1) turned out to hold. This validates the correctness of the simulation model, at least for the given traces. Likewise, in traces where the strategy profile described a preference for arithmetical representations, properties such as "*if possible, only arithmetic representations are used*" are satisfied.

In addition, note that the automated checker can also take empirical reasoning traces as input. Using this approach, in future research it will be checked which properties hold for empirical data, thereby supporting the comparison of human reasoning with simulated reasoning.

Note that the type of checks of dynamic properties on traces that are performed here are not based on the technique in the literature called model checking (e.g., Clarke, Grumberg, and Peled, 2000). To perform model checking of dynamic properties, the state space explosion problem has to be addressed, since in model checking all possible traces have to be explored. However, checking properties not on all possible, but only on one trace, or a given limited set of traces, as done in this paper, is much easier. Therefore, in our case the state space explosion problem does not occur, and checks are easy to perform. Moreover, in this case the language for the dynamic properties can also be more expressive, such as the sorted predicate logic temporal trace language TTL described in this paper, which is much more expressive than the propositional modal logic languages used in model checking.

8. Other Examples of Component-Based Reasoning Models

In this section some other applications of DESIRE to design component-based reasoning models are briefly described. First, a model for reasoning by assumption is briefly described. Next three other reasoning models will be summarized: for nonmonotonic reasoning, for BDI-reasoning, and for normative reasoning.

In (Jonker and Treur, 2003) a component-based model for reasoning by assumption can be found; this will be explained in some more detail. Reasoning by assumption incorporates reasoning with and about assumptions. Reasoning about assumptions can be considered as a form of *meta-reasoning*. The agent reasons about a set of assumptions when deciding for them to be assumed for a while (reasoning *about* assumptions). After making assumptions, the agent derives which facts are logically implied by this set of assumptions (reasoning *with* assumptions). The derived facts may be evaluated; based on this evaluation some of the assumptions may be rejected and/or a new set of assumptions may be chosen (reasoning *about* assumptions). As an example, if an assumption has been chosen, and the facts derived from this assumption contradict information obtained from a different source (e.g., by observation), then the assumption is rejected and the converse assumed.

A generic reasoning model[2] behind this pattern of reasoning has been designed in the component-based design method DESIRE; cf. (Brazier, Jonker, and Treur, 2000, 2002). This formally specified design has been automatically translated into a software program capable of simulating the reasoning process. The reasoning model consists of four basic (primitive) components: External World, Observation Result Prediction, Assumption Determination, and Assumption Evaluation (see Figure 6). The component External World contains the world state and is used to execute observations. The component Observation Result Prediction reasons *with* assumptions. The two components Assumption Determination and Assumption Evaluation reason *about* assumptions (they perform the meta-reasoning). Information is exchanged between the components where necessary.

[2] A complete specification of the model (with clickable components) can be found at: www.cs.vu.nl/~wai/GTM/assumption/assumption_fixed_tc_2WP_07

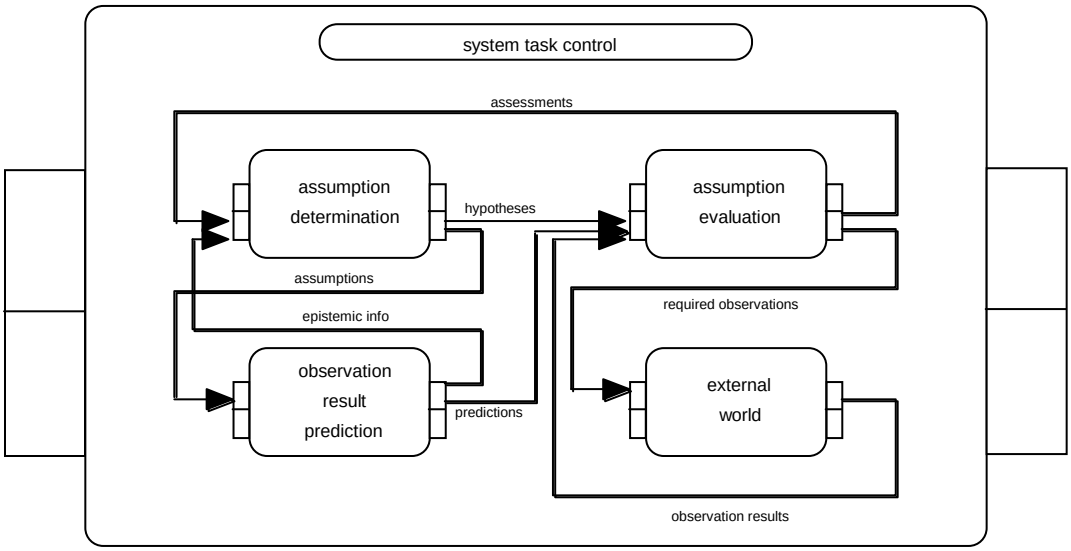

Figure 6. Architecture of the simulation model

The component *Assumption Determination* performs metareasoning to derive which assumption to make. The knowledge specified for this component expresses a form of heuristic knowledge able to generate assumptions for the different situations. It is taken into account whether or not an assumption already has been considere before (i.e., was an assumption earlier), to avoid repetition. The component *Observation Result Prediction* takes an assumption and derives from this assumption what should be expected as observations in the world, for example, using causal knowledge. The component *Assumption Evaluation*, based on generic knowledge, compares predictions and observations, and, where these are conflicting, rejects the underlying assumption; see the first generic rule. A second functionality is to determine which observations have to be made, namely, those for which predictions exist; this is specified in the second generic rule.

```
if assumed(HYP: INFO_ELEMENT, S: SIGN)
   and predicted_for(OBS: INFO_ELEMENT, S1:SIGN,  HYP: INFO_ELEMENT, S: SIGN)
   and observation_result(OBS: INFO_ELEMENT, S2:SIGN)
   and S1 ≠ S2
then   rejected(HYP: INFO_ELEMENT, S: SIGN)
   and has_been_considered(HYP: INFO_ELEMENT, S: SIGN)

if predicted_for(OBS : INFO_ELEMENT, S1: SIGN, HYP: INFO_ELEMENT, S2: SIGN)
then   to_be_observed(OBS : INFO_ELEMENT)
```

The agent reasoning model described here has been used to model reasoning processes based on assumption, such as diagnosis (Brazier, Jonker, Treur, and Wijngaards, 2000), and solving reasoning puzzles (Jonker and Treur, 2003), where also more details can be found.

Furthermore, in (Engelfriet and Treur, 2003) a generic reasoning system is described that can be used to perform nonmotonic reasoning. It can take any nonmotonic logical theory and perform reasoning based on that theory. It is a rather complex component-based reasoning system, which has been described in detail in (Engelfriet and Treur, 2003).

Moreover, in (Brazier, Dunin-Keplicz, Treur, and Verbrugge, 1999), a component-based model for a BDI-agent is described. Main components are Belief Determination, Desire Determination and Intention Determination. For more details, see the reference mentioned.

Finally, in (Castelfranchi, Dignum, Jonker, and Treur, 2000), a component-based model for a normatively reasoning agent is shown. The agent model, built on top of a BDI-model, deliberates about its options and involves normative criteria in the generation and selection of desires and intentions; for more details, see the reference mentioned.

9. Discussion

Analysis of the cognitive capability to perform reasoning has been addressed from different areas and angles. Within Cognitive Science, the two dominant streams are the syntactic approach (based on inference rules applied to syntactic expressions, as common in logic), e.g., (Rips, 1994), and the semantic approach (based on construction of mental models); e.g., (Johnson-Laird, 1983). In experimental work for these approaches reasoning processes usually are studied by focussing on reasoning steps in isolation, by means of one-trial experiments. More extensive reasoning processes involving a number of steps that are tuned to each other require coherent controlled navigation. The current paper reports analysis and simulation of such a reasoning process.

Reasoning with multiple representations can be quite useful, as operations can be available for manipulation of one type of representation but not for another type. In the case study on multiplication addressed here, it is possible to use geometrical operations to find the number that is the product of two given numbers. If a multiplication problem to be solved is offered in arithmetical form, then a number of translations between representations have to be made, from arithmetical to geometrical representation, and back. Such translations are in fact a special case of ontology mappings. For example, numbers a and b to be multiplied are mapped onto line segments AB and AC, or the area of a rectangle ABCD is mapped onto a number c. In Figure 2 in steps 1, 3, 5, 7, 8 such translations take place. In the simulation models such mappings have been built-in. They are part of the domain knowledge. Also in other application domains to be addressed, these ontology mappings have to be acquired to let a model like this work. If they are not available, then the approach cannot work.

The analysis method for the dynamics of reasoning processes used in this paper was adopted from (Jonker and Treur, 2002) and validated on the basis of reports from experiments with 8-9 year old children in classrooms in the Netherlands (Dekker et al., 1982). A similar report has been made by (Hutton, 1977). The current paper shows how an analysis of these dynamics can be made using traces consisting of sequences of reasoning states including control information over time to describe controlled reasoning processes. It is shown for the example reasoning pattern, how characterising dynamic properties can be identified. Furthermore, the agent modelling approach DESIRE has been used to specify and implement a simulation model, and other software tools have been used to automatically check which dynamic properties hold for which simulated traces. In addition, these software tools can be used to check which properties hold for empirical data, thereby supporting the comparison of human reasoning with simulated reasoning. The variety of dynamic properties specified and the variety of traces simulated provides an overview for the individual differences between subjects that have been observed while solving multiplication problems. For example, using our formalisation those with an emphasis on external arithmetic representations are neatly distinguished from those who use external material representations where possible. In the analysis the notion of reasoning strategy was addressed, incorporating such differences in skill and preference. Due to the compositional structure of reasoning state it was not difficult to extend a reasoning state with a component for control information.

Note that the modelling approach used in this paper makes a clear distinction between generic and domain-specific aspects. This makes it relatively easy to plug in a different domain in multi-representational reasoning. For example, the domain can be modified to an example for children of 13 or 14 years to support algebra by geometric visualisations, e.g., the algebraic identity $(a + b)^2 = a^2 + 2ab + b^2$ interpreted as the area of a partitioned square of (a + b) x (a + b) in relation to areas of its parts: a square of a x a, a square of b x b, and two rectangles of a x b.

To make it feasible to perform model checking (e.g., Clarke, Grumberg, and Peled, 2000) of dynamic properties, the state space explosion problem has to be addressed. In the first place, expressivity of the language for these dynamic properties has to be sacrificed to a large extent. However, checking properties on a given set of traces of practical size (instead of all theoretically possible traces), obtained empirically or by simulation, as done in this paper is not full model checking, and therefore computationally much cheaper. Thus, in our case the state space explosion problem does not occur, and the language for the properties to be checked can be more expressive,

such as the sorted predicate logic temporal trace language TTL described in this paper. Note that in this way no logical consequence relations between properties are found but only that properties hold (or not) for the given set of traces. This is a difference not only to model checking but also to theorem proving. The fact that the properties hold for the given set of traces depends on this set and therefore not a general logical theorem as aimed for in model checking and theorem proving. That explains why the algorithm used in our case avoids the combinatiorial problems of both model checking and theorem proving: it has a more modest aim to achieve.

With respect to future work, further experiments will be conducted, in which also a focus is more explicitly on the control of the reasoning. For example, are subjects able to explain why at a point in time a translation to a geometric representation is made? Are think-aloud protocols involving control information a reliable source of further analysis? In addition, future work will explore the possibility to reuse the current simulation model in other cognitive domains.

References

Aridor, Y., and Lange, D.B. (1998). Agent Design Patterns: Elements of Agent Application Design. Proc. of the Second Annual Conference on Autonomous Agents, Agents'98, ACM Press, pp. 108-115.

Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M. (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons, 1996.

Booch, G. (1994). Object-Oriented Analysis and Design (2nd ed.). Addison-wesley. reading, MA.

Booker, G., Bond, D., Briggs, J. & Davey, G. (1997). Teaching Primary Mathematics, Melbourne: Longman.

Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J., (2006). Specification and Verification of Dynamics in Cognitive Agent Models. In: *Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06*. IEEE Computer Society Press, 2006, to appear

Brazier, F.M.T., Cornelissen, F., Gustavsson, R., Jonker, C.M., Lindeberg, O., Polak, B., and Treur, J., (2002). A Multi-Agent System Performing One-to-Many Negotiation for Load Balancing of Electricity Use. *Electronic Commerce Research and Applications Journal*, vol. 1, 2002, pp. 208 - 224.

Brazier, F.M.T., Cornelissen, F., Jonker, C.M., and, Treur, J., (2000). Compositional Specification of a Reusable Co-operative Agent Model *International Journal of Cooperative Information Systems*, vol. 9, 2000, pp. 171-207.

Brazier, F.M.T., Dunin-Keplicz, B.M., Treur, J., and Verbrugge, L.C., (1999). Modelling Internal Dynamic Behaviour of BDI agents. In: J.-J. Ch. Meyer and P.Y. Schobbes (eds.), *Formal Models of Agents (Selected papers from final ModelAge Workshop).* Lecture Notes in AI, vol. 1760, Springer Verlag, 1999, pp. 36-56. Also in: Gabbay, D., and Smets, Ph. (eds.), *Dynamics and Management of Reasoning Processes.* Series in Defeasible Reasoning and Uncertainty Management Systems, vol. 6. Kluwer Academic Publishers, 2001, pp. 339-361.

Brazier, F.M.T., Jonker, C.M., Treur, J., and Wijngaards, N.J.E., (2000). On the Use of Shared Task Models in Knowledge Acquisition, Strategic User Interaction and Clarification Agents. *International Journal of Human-Computer Studies*, vol. 52, 2000, pp. 77-110.

Brazier, F.M.T., Jonker, C.M., and Treur, J., (2000). Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal*, vol. 14, 2000, pp. 491-538.

Brazier, F. M. T., Jonker, C.M., and Treur, J., (2002). Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering, vol. 41, pp. 1-28.

Brazier, F.M.T., Jonker, C.M., and Treur, J., Dynamics and Control in Component-Based Agent Models. *International Journal of Intelligent Systems*, vol. 17, 2002, pp. 1007-1048.

Brazier, F.M.T., and Treur J., (1999). Compositional Modelling of Reflective Agents. *International Journal of Human-Computer Studies*, vol. 50, 1999, pp. 407-431.

Bruner, J.S. (1968). *Toward a Theory of Instruction*. Norton & Company, Inc. New York.

Castelfranchi, C., Dignum, F., Jonker, C.M. and Treur, J., (2000). Deliberative Normative Agents: Principles and Architecture. In: N.R. Jennings, Y. Lesperance (eds.), *Intelligent Agents VI. Proc. of the Sixth International Workshop on Agent Theories, Architectures and Languages, ATAL'99*. Lecture Notes in AI, vol. 1757, Springer Verlag, 2000, pp. 364-378.

Clarke, E.M., Grumberg, O., and Peled, D.A. (2000). Model Checking. MIT Press.

Dekker, A., Heege, H. ter, and Treffers, A. (1982). *Cijferend vermenigvuldigen en delen volgens Wiskobas*. Universiteit Utrecht, Freudenthal Institute.

Engelfriet, J., and Treur, J., (2003). A Compositional Reasoning System for Executing Nonmonotonic Theories of Reasoning. *International Journal of Intelligent Systems*, vol. 18, 2003, pp. 593-608.

English, L. & Halford, G. (1995) Mathematics Education: Models and Processes. Mahwah, NJ: Lawrence Erlbaum

Fisher, M., (1995). Representing and executing agent-based systems. In: Intelligent Agents—Proceedings of the International Workshop on Agent Theories, Architectures, and Languages, ATAL'94, edited by M. Wooldridge and N. Jennings, Lecture Notes in Artificial Intelligence, vol. 890, Springer-Verlag: Berlin, 1995.

Fisher, M., (2005). Temporal Development Methods for Agent-Based Systems, *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 10, 2005, pp. 41-66.

Galton, A. (2003). Temporal Logic. Stanford Encyclopedia of Philosophy, URL: http://plato.stanford.edu/entries/logic-temporal/#2

Galton, A. (2006). Operators vs Arguments: The Ins and Outs of Reification. *Synthese*, vol. 150, (2006), pp. 415-441.

Gamma, EE., Helm, R., Johnson, R. and Vlissides, J., (1994). *Desing Patterns: Elements of reusable object-oriented Software*, Addison Wesley Longman, Reading, Massachusetts, 1994.

Hegarty, M. (2002). Mental Visualizations and External Visualizations. In: W.D. Gray and C.D. Schunn (eds.), Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci'02. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, p. 40.

Herlea Damian, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E., (2005). Integration of Behavioural Requirements Specification within Compositional Knowledge Engineering. *Knowledge-Based Systems Journal*, vol. 18, 2005, pp. 353-365.

Hölldobler, S., and Thielscher, M. (1990). A new deductive approach to planning. *New Generation Computing*, 8:225-244, 1990.

Hutton, J. (1977). Memoirs of a Maths Teacher 5: Logical Reasoning. In: *Mathematics Teaching*, vol. 81, pp. 8-12.

Jennings, N.R., and M. Wooldridge (1998a), Applications of Intelligent Agents. In: Jennings, N.R., and M. Wooldridge (eds.) (1998b), *Agent Technology: Foundations, Applications, and Markets*. Springer Verlag, pp. 3-28.

Johnson-Laird, P.N. (1983). *Mental Models*. Cambridge: Cambridge University Press.

Jonker, C.M., and Treur, J., (2003). Modelling the Dynamics of Reasoning Processes: Reasoning by Assumption. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 119-136.

Jonker, C.M., and Treur, J. (2002). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92. Preliminary, shorter version in: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), *Proceedings of the International Workshop on Compositionality, COMPOS'97*. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380. Extended version in:

Koedinger, K.R., and Terao, A. (2002). A Cognitive Task Analysis of Using Pictures To Support Pre-Algebraic Reasoning. In: W.D. Gray and C.D. Schunn (eds.), Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci'02. Mahwah, NJ: Lawrence Erlbaum Associates, 2002, pp. 542-547.

Kowalski, R., and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67-95, 1986.

Mulder, M., Treur, J., and Fisher, M., (1998). Agent Modelling in MetateM and DESIRE. In: M.P. Singh, A.S. Rao, M.J. Wooldridge (eds.), Intelligent Agents IV, Proc. Fourth International Workshop on Agent Theories, Architectures and Languages, ATAL'97. Lecture Notes in AI, vol. 1365, Springer Verlag, 1998, pp. 193-207.

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

Rips, L.J. (1994). *The Psychology of Proof: Deductive reasoning in human thinking*. MIT Pres, Cambridge, Mass.

Shehory, O., and Sturm, A. (2001). Evaluation of modeling techniques for agent-based systems. In: Jorg P. Muller, Elisabeth Andre, Sandip Sen, and Claude Frasson, (eds.), *Proceedings of the Fifth International Conference on Autonomous Agents*. ACM Press, 2001, pp. 624-631

Wooldridge, M., Jennings, N.R. (eds.) (1995a) *Intelligent Agents*, Lecture Notes in Artificial Intelligence, Vol. 890, Springer Verlag, Berlin,

Wooldridge, M.J., Jennings, N.R. (1995b) Agent theories, architectures, and languages: a survey. In: (Wooldridge and Jennings, 1995a), pp. 1-39.

Wooldridge, M.J., and N.R. Jennings (1995c). Intelligent Agents: Theory and practice. In: *Knowledge Engineering Review*, 10(2), pp. 115-152.

Yang, Y., and Johnson-Laird, P.N. (1999). A study of complex reasoning: The case GRE 'logical' problems. In M. A. Gernsbacher & S. J. Derry (Eds.) *Proceedings of the Twenty First Annual Conference of the Cognitive Science Society*, 767-771.

Appendix A. Simulation Traces

This Appendix contains a number of simulation traces that were generated on the basis of the model described in Section 5.

Trace 1

Step	Information Derived	
0	strategy profile:	geo-ari-mat
0	available abstract skills:	all skills
0	available mental skills:	all skills
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
2	plan([bs1, bs7, bs2, bs4, bs5, bs9, bs3, bs6, bs13, bs14])	
3	is_represented_in_world(arithmetic, multiplication(23, 36))	
4	mental_representation(geometric, entity(shape("[]"), parameters(23, 36)))	
5	is_represented_in_world(geometric, rectangle('A', 'B', 'C', 'D', 23, 36))	
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(36, 30, 6)))	
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(23, 20, 3)))	
7	is_represented_in_world(arithmetic, split(36, 30, 6))	
7	is_represented_in_world(arithmetic, split(23, 20, 3))	
8	mental_representation(geometric, entity(shape("-"), name('A', 'B'), parameters(20, 3)))	
8	mental_representation(geometric, entity(shape("-"), name('A', 'D'), parameters(30, 6)))	
9	is_represented_in_world(geometric, split('A', 'B', 20, 3))	
9	is_represented_in_world(geometric, split('A', 'D', 30, 6))	
10	mental_representation(geometric, entity(shape("[]"), name('A11'), parameters(20, 30)))	
10	mental_representation(geometric, entity(shape("[]"), name('A12'), parameters(20, 6)))	
10	mental_representation(geometric, entity(shape("[]"), name('A21'), parameters(3, 30)))	
10	mental_representation(geometric, entity(shape("[]"), name('A22'), parameters(3, 6)))	
11	is_represented_in_world(geometric, area('A11', 20, 30))	
11	is_represented_in_world(geometric, area('A12', 20, 6))	
11	is_represented_in_world(geometric, area('A21', 3, 30))	
11	is_represented_in_world(geometric, area('A22', 3, 6))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 6)))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 30)))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 6)))	
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 30)))	
13	is_represented_in_world(arithmetic, partial_multiplication('A11', 20, 30))	
13	is_represented_in_world(arithmetic, partial_multiplication('A12', 20, 6))	
13	is_represented_in_world(arithmetic, partial_multiplication('A21', 3, 30))	
13	is_represented_in_world(arithmetic, partial_multiplication('A22', 3, 6))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 6, 18)))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 30, 90)))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 6, 120)))	
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 30, 600)))	
15	is_represented_in_world(arithmetic, multiplication_solution(3, 6, 18))	
15	is_represented_in_world(arithmetic, multiplication_solution(3, 30, 90))	
15	is_represented_in_world(arithmetic, multiplication_solution(20, 6, 120))	
15	is_represented_in_world(arithmetic, multiplication_solution(20, 30, 600))	
16	mental_representation(geometric, entity(shape("[]"), name('A11'), area_with_number(600)))	
16	mental_representation(geometric, entity(shape("[]"), name('A12'), area_with_number(120)))	
16	mental_representation(geometric, entity(shape("[]"), name('A21'), area_with_number(90)))	
16	mental_representation(geometric, entity(shape("[]"), name('A22'), area_with_number(18)))	
17	is_represented_in_world(geometric, area_with_number('A11', 600))	
17	is_represented_in_world(geometric, area_with_number('A12', 120))	
17	is_represented_in_world(geometric, area_with_number('A21', 90))	
17	is_represented_in_world(geometric, area_with_number('A22', 18))	
18	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))	
19	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))	
20	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))	
21	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))	
22	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))	
23	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))	

Trace 2

Step	Information Derived	
0	strategy profile:	ari-mat-geo
0	available abstract skills:	all skills except bs10
0	available mental skills:	all skills except bs10
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
2	plan([bs24, bs25, bs9, bs26, bs13, bs14])	
3	is_represented_in_world(arithmetic, multiplication(23, 36))	
4	mental_representation(arithmetic, entity(shape("XY*"), parameters(23, 36)))	
5	is_represented_in_world(arithmetic, symbolic_multiplication(23, 36))	
6	mental_representation(arithmetic, entity(shape("X*Y"), parameters(2, 36)))	
6	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 36)))	

7	is_represented_in_world(arithmetic, partial_multiplication('A12', 3, 36))
7	is_represented_in_world(arithmetic, partial_multiplication('A11', 2, 36))
8	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(2, 36, 72)))
8	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 36, 108)))
9	is_represented_in_world(arithmetic, multiplication_solution(2, 36, 72))
9	is_represented_in_world(arithmetic, multiplication_solution(3, 36, 108))
10	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(108, 720, 0, 0)))
11	is_represented_in_world(arithmetic, addition(108, 720, 0, 0))
12	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(108, 720, 0, 0, 828)))
13	is_represented_in_world(arithmetic, addition_solution(108, 720, 0, 0, 828))
14	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
15	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 3

Step	Information Derived	
0	strategy profile:	mat-geo-ari
0	available abstract skills:	all skills
0	available mental skills:	all skills
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
2	is_represented_in_world(arithmetic, multiplication(23, 36))	
3	mental_representation(material, entity(shape("[]"), parameters(23, 36)))	
3	mental_representation(material, entity(shape("[]"), size(big), number(0)))	
3	mental_representation(material, entity(shape("[]"), size(medium_h), number(0)))	
3	mental_representation(material, entity(shape("[]"), size(medium_v), number(0)))	
3	mental_representation(material, entity(shape("[]"), size(small), number(0)))	
4	is_represented_in_world(material, rectangle('A', 'B', 'C', 'D', 23, 36))	
4	is_represented_in_world(material, block(big, 0))	
4	is_represented_in_world(material, block(medium_h, 0))	
4	is_represented_in_world(material, block(medium_v, 0))	
4	is_represented_in_world(material, block(small, 0))	
5	mental_representation(material, entity(shape("[]"), size(big), number(1)))	
6	is_represented_in_world(material, block(big, 1))	
7	mental_representation(material, entity(shape("[]"), size(big), number(2)))	
8	is_represented_in_world(material, block(big, 2))	
9	mental_representation(material, entity(shape("[]"), size(big), number(3)))	
10	is_represented_in_world(material, block(big, 3))	
11	mental_representation(material, entity(shape("[]"), size(big), number(4)))	
12	is_represented_in_world(material, block(big, 4))	
13	mental_representation(material, entity(shape("[]"), size(big), number(5)))	
14	is_represented_in_world(material, block(big, 5))	
15	mental_representation(material, entity(shape("[]"), size(big), number(6)))	
16	is_represented_in_world(material, block(big, 6))	
17	mental_representation(material, entity(shape("[]"), size(big), total(6)))	
18	is_represented_in_world(material, total_of_blocks(big, 6))	
19	mental_representation(material, entity(shape("[]"), size(medium_h), number(1)))	
20	is_represented_in_world(material, block(medium_h, 1))	
21	mental_representation(material, entity(shape("[]"), size(medium_h), number(2)))	
22	is_represented_in_world(material, block(medium_h, 2))	
23	mental_representation(material, entity(shape("[]"), size(medium_h), number(3)))	
24	is_represented_in_world(material, block(medium_h, 3))	
25	mental_representation(material, entity(shape("[]"), size(medium_h), number(4)))	
26	is_represented_in_world(material, block(medium_h, 4))	
27	mental_representation(material, entity(shape("[]"), size(medium_h), number(5)))	
28	is_represented_in_world(material, block(medium_h, 5))	
29	mental_representation(material, entity(shape("[]"), size(medium_h), number(6)))	
30	is_represented_in_world(material, block(medium_h, 6))	
31	mental_representation(material, entity(shape("[]"), size(medium_h), number(7)))	
32	is_represented_in_world(material, block(medium_h, 7))	
33	mental_representation(material, entity(shape("[]"), size(medium_h), number(8)))	
34	is_represented_in_world(material, block(medium_h, 8))	
35	mental_representation(material, entity(shape("[]"), size(medium_h), number(9)))	
36	is_represented_in_world(material, block(medium_h, 9))	
37	mental_representation(material, entity(shape("[]"), size(medium_h), number(10)))	
38	is_represented_in_world(material, block(medium_h, 10))	
39	mental_representation(material, entity(shape("[]"), size(medium_h), number(11)))	
40	is_represented_in_world(material, block(medium_h, 11))	
41	mental_representation(material, entity(shape("[]"), size(medium_h), number(12)))	
42	is_represented_in_world(material, block(medium_h, 12))	
43	mental_representation(material, entity(shape("[]"), size(medium_h), total(12)))	
44	is_represented_in_world(material, total_of_blocks(medium_h, 12))	
45	mental_representation(material, entity(shape("[]"), size(medium_v), number(1)))	
46	is_represented_in_world(material, block(medium_v, 1))	
47	mental_representation(material, entity(shape("[]"), size(medium_v), number(2)))	
48	is_represented_in_world(material, block(medium_v, 2))	
49	mental_representation(material, entity(shape("[]"), size(medium_v), number(3)))	
50	is_represented_in_world(material, block(medium_v, 3))	
51	mental_representation(material, entity(shape("[]"), size(medium_v), number(4)))	
52	is_represented_in_world(material, block(medium_v, 4))	
53	mental_representation(material, entity(shape("[]"), size(medium_v), number(5)))	
54	is_represented_in_world(material, block(medium_v, 5))	
55	mental_representation(material, entity(shape("[]"), size(medium_v), number(6)))	
56	is_represented_in_world(material, block(medium_v, 6))	
57	mental_representation(material, entity(shape("[]"), size(medium_v), number(7)))	
58	is_represented_in_world(material, block(medium_v, 7))	
59	mental_representation(material, entity(shape("[]"), size(medium_v), number(8)))	
60	is_represented_in_world(material, block(medium_v, 8))	

61	mental_representation(material, entity(shape("[]"), size(medium_v), number(9)))
62	is_represented_in_world(material, block(medium_v, 9))
63	mental_representation(material, entity(shape("[]"), size(medium_v), total(9)))
64	is_represented_in_world(material, total_of_blocks(medium_v, 9))
65	mental_representation(material, entity(shape("[]"), size(small), number(1)))
66	is_represented_in_world(material, block(small, 1))
67	mental_representation(material, entity(shape("[]"), size(small), number(2)))
68	is_represented_in_world(material, block(small, 2))
69	mental_representation(material, entity(shape("[]"), size(small), number(3)))
70	is_represented_in_world(material, block(small, 3))
71	mental_representation(material, entity(shape("[]"), size(small), number(4)))
72	is_represented_in_world(material, block(small, 4))
73	mental_representation(material, entity(shape("[]"), size(small), number(5)))
74	is_represented_in_world(material, block(small, 5))
75	mental_representation(material, entity(shape("[]"), size(small), number(6)))
76	is_represented_in_world(material, block(small, 6))
77	mental_representation(material, entity(shape("[]"), size(small), number(7)))
78	is_represented_in_world(material, block(small, 7))
79	mental_representation(material, entity(shape("[]"), size(small), number(8)))
80	is_represented_in_world(material, block(small, 8))
81	mental_representation(material, entity(shape("[]"), size(small), number(9)))
82	is_represented_in_world(material, block(small, 9))
83	mental_representation(material, entity(shape("[]"), size(small), number(10)))
84	is_represented_in_world(material, block(small, 10))
85	mental_representation(material, entity(shape("[]"), size(small), number(11)))
86	is_represented_in_world(material, block(small, 11))
87	mental_representation(material, entity(shape("[]"), size(small), number(12)))
88	is_represented_in_world(material, block(small, 12))
89	mental_representation(material, entity(shape("[]"), size(small), number(13)))
90	is_represented_in_world(material, block(small, 13))
91	mental_representation(material, entity(shape("[]"), size(small), number(14)))
92	is_represented_in_world(material, block(small, 14))
93	mental_representation(material, entity(shape("[]"), size(small), number(15)))
94	is_represented_in_world(material, block(small, 15))
95	mental_representation(material, entity(shape("[]"), size(small), number(16)))
96	is_represented_in_world(material, block(small, 16))
97	mental_representation(material, entity(shape("[]"), size(small), number(17)))
98	is_represented_in_world(material, block(small, 17))
99	mental_representation(material, entity(shape("[]"), size(small), number(18)))
100	is_represented_in_world(material, block(small, 18))
101	mental_representation(material, entity(shape("[]"), size(small), total(18)))
102	is_represented_in_world(material, total_of_blocks(small, 18))
103	mental_representation(arithmetic, entity(shape("X*Y"), parameters(18, 1)))
104	is_represented_in_world(arithmetic, partial_multiplication('A22', 18, 1))
105	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(18, 1, 18)))
106	is_represented_in_world(arithmetic, multiplication_solution(18, 1, 18))
107	mental_representation(material, entity(shape("[]"), size(small), area_with_number(18)))
108	is_represented_in_world(material, area_with_number(small, 18))
109	mental_representation(arithmetic, entity(shape("X*Y"), parameters(9, 10)))
110	is_represented_in_world(arithmetic, partial_multiplication('A21', 9, 10))
111	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(9, 10, 90)))
112	is_represented_in_world(arithmetic, multiplication_solution(9, 10, 90))
113	mental_representation(material, entity(shape("[]"), size(medium_v), area_with_number(90)))
114	is_represented_in_world(material, area_with_number(medium_v, 90))
115	mental_representation(arithmetic, entity(shape("X*Y"), parameters(12, 10)))
116	is_represented_in_world(arithmetic, partial_multiplication('A12', 12, 10))
117	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(12, 10, 120)))
118	is_represented_in_world(arithmetic, multiplication_solution(12, 10, 120))
119	mental_representation(material, entity(shape("[]"), size(medium_h), area_with_number(120)))
120	is_represented_in_world(material, area_with_number(medium_h, 120))
121	mental_representation(arithmetic, entity(shape("X*Y"), parameters(6, 100)))
122	is_represented_in_world(arithmetic, partial_multiplication('A22', 6, 100))
123	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(6, 100, 600)))
124	is_represented_in_world(arithmetic, multiplication_solution(6, 100, 600))
125	mental_representation(material, entity(shape("[]"), size(big), area_with_number(600)))
126	is_represented_in_world(material, area_with_number(big, 600))
127	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))
128	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))
129	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))
130	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))
131	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
132	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 4

Step	Information Derived	
0	strategy profile:	ari-geo-mat
0	available abstract skills:	all skills except bs10
0	available mental skills:	all skills except bs10 and bs25
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
2	plan([bs24, bs25, bs9, bs26, bs13, bs14])	
3	is_represented_in_world(arithmetic, multiplication(23, 36))	
4	mental_representation(arithmetic, entity(shape("XY*"), parameters(23, 36)))	
5	is_represented_in_world(arithmetic, symbolic_multiplication(23, 36))	
	/* mental part of bs25 fails -> create new plan */	
6	plan([bs7, bs1, bs2, bs4, bs5, bs9, bs3, bs6, bs13, bs14])	
7	is_represented_in_world(arithmetic, multiplication(23, 36))	

8	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(36, 30, 6)))
8	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(23, 20, 3)))
9	is_represented_in_world(arithmetic, split(36, 30, 6))
9	is_represented_in_world(arithmetic, split(23, 20, 3))
10	mental_representation(geometric, entity(shape("[]"), parameters(23, 36)))
11	is_represented_in_world(geometric, rectangle('A', 'B', 'C', 'D', 23, 36))
12	mental_representation(geometric, entity(shape("-"), name('A', 'B'), parameters(20, 3)))
12	mental_representation(geometric, entity(shape("-"), name('A', 'D'), parameters(30, 6)))
13	is_represented_in_world(geometric, split('A', 'B', 20, 3))
13	is_represented_in_world(geometric, split('A', 'D', 30, 6))
14	mental_representation(geometric, entity(shape("[]"), name('A11'), parameters(20, 30)))
14	mental_representation(geometric, entity(shape("[]"), name('A12'), parameters(20, 6)))
14	mental_representation(geometric, entity(shape("[]"), name('A21'), parameters(3, 30)))
14	mental_representation(geometric, entity(shape("[]"), name('A22'), parameters(3, 6)))
15	is_represented_in_world(geometric, area('A11', 20, 30))
15	is_represented_in_world(geometric, area('A12', 20, 6))
15	is_represented_in_world(geometric, area('A21', 3, 30))
15	is_represented_in_world(geometric, area('A22', 3, 6))
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 6)))
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 30)))
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 6)))
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 30)))
17	is_represented_in_world(arithmetic, partial_multiplication('A11', 20, 30))
17	is_represented_in_world(arithmetic, partial_multiplication('A12', 20, 6))
17	is_represented_in_world(arithmetic, partial_multiplication('A21', 3, 30))
17	is_represented_in_world(arithmetic, partial_multiplication('A22', 3, 6))
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 6, 18)))
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 30, 90)))
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 6, 120)))
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 30, 600)))
19	is_represented_in_world(arithmetic, multiplication_solution(3, 6, 18))
19	is_represented_in_world(arithmetic, multiplication_solution(3, 30, 90))
19	is_represented_in_world(arithmetic, multiplication_solution(20, 6, 120))
19	is_represented_in_world(arithmetic, multiplication_solution(20, 30, 600))
20	mental_representation(geometric, entity(shape("[]"), name('A11'), area_with_number(600)))
20	mental_representation(geometric, entity(shape("[]"), name('A12'), area_with_number(120)))
20	mental_representation(geometric, entity(shape("[]"), name('A21'), area_with_number(90)))
20	mental_representation(geometric, entity(shape("[]"), name('A22'), area_with_number(18)))
21	is_represented_in_world(geometric, area_with_number('A11', 600))
21	is_represented_in_world(geometric, area_with_number('A12', 120))
21	is_represented_in_world(geometric, area_with_number('A21', 90))
21	is_represented_in_world(geometric, area_with_number('A22', 18))
22	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))
23	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))
24	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))
25	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))
26	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
27	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 5

Step	Information Derived	
0	strategy profile:	ari-geo-mat
0	available abstract skills:	all skills except bs7, bs15, bs24
0	available mental skills:	all skills
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
	/* no further derivations: agent fails to make a plan */	