

## University of Groningen

### The social cognitive actor

Helmhout, M.

**IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.**

*Document Version*

Publisher's PDF, also known as Version of record

*Publication date:*

2006

[Link to publication in University of Groningen/UMCG research database](#)

*Citation for published version (APA):*

Helmhout, M. (2006). *The social cognitive actor: a multi-actor simulation of organisations*. [Thesis fully internal (DIV), University of Groningen]. s.n.

**Copyright**

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

**Take-down policy**

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

*Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.*

## APPENDIX A

### Addition Productions\*

<b>1</b>	<b>Start-Problem push goal check declarative memory</b>	
	<b>IF</b>	- the goal is to do an addition problem
	<b>AND</b>	- status is start and column is empty
	<b>THEN</b>	- set the current goal status to "after checking declarative memory" - push a new subgoal chunktype check.declarative to retrieve the complete answer to the addition from memory
<b>2</b>	<b>Check addition problem in memory</b>	
	<b>IF</b>	- the goal is to retrieve the answer from memory
	<b>AND</b>	- the answer to the problem is found in memory
	<b>THEN</b>	- substitute the answer in a chunk and put it in the communication buffer - remove the subgoal - set the upper goal status to found
<b>3</b>	<b>Found and clean</b>	
	<b>IF</b>	- the goal has a status of found
	<b>THEN</b>	- remove the top goal <i>(end cycle solved problem)</i>
<b>4</b>	<b>After check declarative memory start problem</b>	
	<b>IF</b>	- the goal has status "after checking declarative memory"
	<b>THEN</b>	- set goal status to start and column = 1
<b>5</b>	<b>Addition read goal</b>	
	<b>IF</b>	- status equals start and column == 1 Remember temporary variables col1number1 = 5 col1number2 = 7 nrColumns = 4
	<b>THEN</b>	- set goal status to wait for subgoal - transfer nrColumns=4 to retrievalbuffer - push subgoal chunktype add.column number1 5 number2 7 column 1 carry Zero

\*Productions of the multi-column addition problem in English Language

Appendix A: Addition Productions

<b>6</b>	<b>Add numbers no carry</b>	
	<b>IF</b>	- the goal is to add two number with no carry number1 = 5 number2 = 7 carry = Zero
	<b>AND</b>	- if the summation fact (5 + 7=?) can be retrieved from memory
	<b>THEN</b>	- set the answer of add_column to the answer of the summation fact (12) - set the status(carry) on waiting
<b>7</b>	<b>Add numbers process carry</b>	
	<b>IF</b>	- the goal is to add two numbers with a carry number1 = 5 number2 = 7 carry = one
	<b>AND</b>	- if the summation fact (number1 + carry = newnum1 (5+1 = ?)) can be retrieved from memory
	<b>AND</b>	- if the summation fact (newnum1 + number2 = answer (6+7 = ?)) can be retrieved from memory
	<b>THEN</b>	- set the answer of add_column to the answer of the summation fact(13) - set the status(carry) on waiting
<b>8</b>	<b>Add column answer greater than 9 (creating a carry)</b>	
	<b>IF</b>	- the goal is to find out if there is a carry after addition number1 = 5 number2 = 7 answer = 12 carry == waiting
	<b>AND</b>	- it is not the last column (column != nrColumns)
	<b>AND</b>	- the summation fact (10 + addend2 = answer (10 + 2 = 12)) can be retrieved from memory
	<b>THEN</b>	- set the answer to addend2 (2) and carry to one - remove nrColumns from retrieval buffer - put current goal in retrieval buffer - pop the goal with the add_column
<b>9</b>	<b>Add column answer smaller than 10 (no carry)</b>	
	<b>IF</b>	- the goal is to find out if there is no carry after addition number1 = 2 number2 = 7 answer = 9 carry == waiting
	<b>AND</b>	- it is not the last column (column != nrColumns)
	<b>AND</b>	- the summation fact (0 + addend2 = answer (0 + 9 = 9)) can be retrieved from memory
	<b>THEN</b>	- set the answer to addend2 (9) and carry to zero - remove nrColumns from retrieval buffer - put current goal in retrieval buffer - pop the goal with the add_column (back at top goal == addition-problem)
<b>10</b>	<b>Add column last column</b>	
	<b>IF</b>	the goal is to add the last column of the addition problem carry == waiting
	<b>AND</b>	- it is the last column (column equals nrColumns) - set carry to "finished"
	<b>THEN</b>	- remove nrColumns from retrieval buffer - put current goal in retrieval buffer - pop the current goal with the add_column

## Appendix A: Addition Productions

11	<b>Addition problem write column abstract</b> ( <i>depending on column number</i> )	
	IF	<ul style="list-style-type: none"> <li>the goal is to write the answer to the goal chunk</li> <li>column !=null</li> <li>nrColumns != column</li> <li>column is 1... 4 (column number)</li> <li>col(column number)answer == null</li> </ul>
	AND	- in the retrieval buffer is a chunk of add_column
	AND	- retrieve (1+ column number = new column number (e.g. 2 → 3) from memory
	AND	- read the new numbers of the new column
	THEN	<ul style="list-style-type: none"> <li>- write the answer to the col(column number)answer slot of the goal</li> <li>- remove the chunk from retrieval buffer and put nrColumns in retrieval buffer</li> <li>- push the new subgoal with a new add_column of the new column</li> </ul>
12	<b>Addition problem write last column</b>	
	IF	<ul style="list-style-type: none"> <li>- the goal is to write the answer of the last column to the goal chunk</li> <li>nrColumns equals column number</li> <li>status equals not communicate</li> </ul>
	AND	- if there is an add_column chunk in the retrieval buffer
	THEN	<ul style="list-style-type: none"> <li>- write the answer to the last column</li> <li>- set status to communicate</li> <li>- remove the chunk from the retrieval buffer</li> </ul>
13	<b>Addition problem communicate</b>	
	IF	- status is communicate
	THEN	<ul style="list-style-type: none"> <li>- put goal chunk in communication buffer</li> <li>- put the chunk in declarative memory</li> <li>- pop the goal : problem solved</li> </ul>

