Herbal Tutorial

Frank E. Ritter¹, Geoff Morgan¹, Bill Stevenson¹, Mark A. Cohen¹,²

1 School of IST, Penn State
2 BA CS & IT Department, Lock Haven University

What is this tutorial?/Who is it for?

- Assumes some Soar knowledge
  ➢ this can be fixed, or may not be necessary
- Interested in high-level modeling/AI languages
- Students of architecture
- Willing to learn Protege
- GSOH

This project was supported by the US Office of Naval Research, award number N000140210021. We also thank Steve Haynes, Geoffrey P. Morgan, Urmila Kukreja, Isaac Councill, and the students of IST 402 the last two years for their work with and comments on Herbal. Discussions with AOS, Harold Hawkins, Randy Jones, Robert St. Amant, Glenn Taylor, and Bob Wray have informed this work. Josh Gross, Sue Kase, and Urmila Kukreja has helped test Herbal.

The views expressed in this article do not necessarily reflect the positions or the policies of the U.S. Government, and no official endorsement should be inferred.
Today’s Activities

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to tutorial</td>
<td>~10 mins</td>
</tr>
<tr>
<td>Overview of Herbal</td>
<td>~20 mins</td>
</tr>
<tr>
<td>Protégé tutorial</td>
<td>~40 mins</td>
</tr>
<tr>
<td>Break</td>
<td>~10 mins sharp</td>
</tr>
<tr>
<td>Blocks World Model</td>
<td>~35 mins</td>
</tr>
<tr>
<td>Blocks World with Tie</td>
<td>~15 mins</td>
</tr>
<tr>
<td>dTank Model</td>
<td>~20 mins</td>
</tr>
<tr>
<td>Viewer</td>
<td>~10 mins</td>
</tr>
<tr>
<td>Wrap-up</td>
<td>~20 mins</td>
</tr>
</tbody>
</table>

Introducing the Tutors

- Frank Ritter, cognitive modeler, interested in learning, interaction, moderators, and methods
- Geoff Morgan, recent graduate, used Herbal in IST 402
- Bill Stevenson, TA for IST 402
- Mark Cohen, current programmer and co-designer

- Get software installed by/with them during introduction segments
Introduction

Soar on a Slide 1 of 2

Production Memory (LTM)

\[ A \land B \Rightarrow X \]
\[ C \land D \Rightarrow Y \]
\[ \ldots \ldots \Rightarrow \ldots \]

conditions test SDM

actions write into SDM

new “results” give rise to new productions

Soar’s Working Memory (WM)

block

green

impasse

operator: paint S1

operator no-change S2

Taken from the Soar video (1994)
acs.ist.psu.edu/papers/soar-mov.mpg

Soar on a Slide 2 of 2

Introduction
Herbal—A High-Level Language and Development Environment for Developing Cognitive Models in Soar

Mark A. Cohen\textsuperscript{1,2}, Frank E. Ritter\textsuperscript{2}, Steve Haynes\textsuperscript{2}

1 BA CS & IT Department, Lock Haven University
2 School of IST, Penn State

This project was supported by the US Office of Naval Research, award number N000140210021. We also thank Geoffrey P. Morgan, Urmila Kukreja, Isaac Councill, and the students of IST 402 the last two years for their work with and comments on Herbal.

The views expressed in this article do not necessarily reflect the positions or the policies of the U.S. Government, and no official endorsement should be inferred.

A Problem and Solution (Ritter et al., 2003)

- Cognitive architecture models appear like assembly code
- Explanation
- Reuse
- Ease of use

A high-level language that maps more directly to the domain the user is familiar with
Existing Solutions

- Visual Soar
- SoarDoc
- ViSoar
- G2A
- All graphical IDEs, e.g., JACK, iGen, Cogent

What Users Want

Based on 4 expert SAP users (Councill et al., 2003)

Queries (111)
- Increased Access to Agent Knowledge (35)
- Queries of agent state (36)
- Usefulness of Existing Knowledge Access Tools (33)
- Lack of Information (7)
- Milestones (9)
- Goal Stack (17)
- Clarity (4)
- Content (5)
- Clarity (4)
- Content (5)
- Superfluous (5)

Summary: Better description of procedural knowledge needed
Using a High Level Behavioral Representation Language

- Design rationale anchors model explanations
  - Design knowledge capture during development
- Model description needed for explanations (Haynes, 2003)
  - Create model parts within IDE
- Responsibility-driven approach through a compiler
  - Organize knowledge and rules
    - Haynes et al., 2004; St. Amant & Ritter, 2004, www4.ncsu.edu/~stamant/G2A

Herbal Design

- Augment existing planning language with design rationale
  - Chose PSCM, RDF, and Protoge: tool availability & generality
  - Also studied direct translation (St. Amant & Ritter, 2004)
  - Ontology also helps structure
- Explanation from declarative representation + rationale
- Compile into Soar rules (XSLT)
  - Could also compile into ACT-R, JACK?
- Designed to leverage VISTA
  - Declarative representation supports model tracing
    - acs.ist.psu.edu/vista for our local training examples
Output:

- Task
- Port Chart (CPM-GOMS)
- ACT-R Rules
- Declarative Memory
- Operators
  - State Augmentations
  - Explanation Knowledge
- Eye-hand
- d'Tank Sim.

Herbal Integrated Development Environment (IDE) - Overview
Herbal Viewer with Blocks World

Herbal IDE - Model tab Overview

The purpose of this model is to arrange three blocks on a table in a specified order:

- Name: Blocks World Model
- Original Author: Mark A. Cohen
- Creation Date: 06/01/2004
- Last Updated: 05/06/2005
- Purpose:
  - The purpose of this model is to arrange three blocks on a table in a specified order.
- How it works:
- Design Constraints:
  - None
The developer can describe the intent of the object.

This descriptive information is embedded in the generated code as comments when the model is compiled.

Will be used in next version to generate explanations (June 05).
Support in Herbal for Explanation

<table>
<thead>
<tr>
<th>Question Type</th>
<th>Support (partial)</th>
<th>Question Type</th>
<th>Support (partial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>What</td>
<td></td>
<td>Functional Explanation</td>
<td>planned IDE</td>
</tr>
<tr>
<td>Identity</td>
<td>Viewer IDE</td>
<td>Pragmatic Explanation (contrast classes)</td>
<td>planned</td>
</tr>
<tr>
<td>Definition</td>
<td>Viewer IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relation</td>
<td>Viewer IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event (what happened?)</td>
<td>Viewer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>How</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How do I use it? - tool</td>
<td>Viewer IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- model</td>
<td>planned IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>How does it work?</td>
<td>Viewer IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Where</td>
<td>Viewer IDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>When</td>
<td>Viewer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Support in Herbal Viewer for Explanation

Planned
How do I use it?
it = <IDE Model>
How does it work?
- Contrast classes
Manual
How do I use it?

What
- identity
- structure

Definition
- structure
+ IDE

What
- relation

When
- order
- how often
Use and Reuse

- Herbal supports reuse within the same model and across models
- Operators and elaborations can share conditions and actions; states can share operators, elaborations, and impasses
- Marginal cost (N=1)
  - (Morgan et al. 2005)

Herbal: Users to Use, Test, Expand Herbal & dTank

- IST 402: Models of human behaviour (12+3, and 38+3 students)
- Microworld to understand, create, and exercise adversarial Soar models
- Class projects
  - MacSoar 8.5 compilation (see Soar web site)
  - Architectural comparisons (Sun et al., 2004)
  - Example reuse curves (Morgan et al., 2005)
  - Errors and Soar (Ritter et al., in prep.)
- In general, the students encountered fewer problems programming in Soar when using Herbal
Experiences Using Herbal (cont.)

- In 2005 9 group final projects were created, and 4/9 used Herbal to create their final projects (all groups used Herbal in their homework).
- 2/4 Herbal teams had a model that ran, and 2/4 of just-Soar groups had a model that ran.
- The most complex working model was an Herbal model with 16 pages of rules with 25 operators.
- All students included a code headers to hook up the models to dTank.

Experiences Using Herbal (cont.)

- Students noted Herbal’s ability to reuse conditions and actions across operators. They also commented on the usefulness the automatically generated comments.
- Herbal did not support impasses then and this was noted by the students as one of the major disadvantages.
  - Impasses added in v. 0.9.
- Students noted usability issues with Protégé.
  - Protégé (version 3.0) addressed many of these issues.
Further Benefits

• Help maintainability
• Ease programming
  • 3x Productivity (Yost, 3 min./production)
  • 1x Productivity (IST 402 class, no complaints of Soar)
  • 1x-2x (Kukreja, first two programs @ 10 hours/program)
  • 5-10-30x (St. Amant, reanalysis of ACT-R model, another phone)
  • 3x Productivity (Morgan et al., 2005, 3 min./production)
• Promote reuse (document and import models)
  (Morgan et al., 2005)
• Explanations may increase learning and use (not mentioned by our subjects)

More Articulate and More Understandable Soar

• High-level language IDE designed to support explanation from declarative representation + rationale
  • With multi-media delivery
  • Improved developer productivity and wider range of outputs
• Micro-world for exploring these issues
  • Audience of users available
  • Architectural and process comparisons are possible
Why Will Herbal Work?

- Principled design based on a theory of knowledge (PSCM, roughly and extended)
- Based on theory of explanations
- Based on data on explanations study
- New payoff - explanations
- Software engineering principles
  - Modularity
  - Software reuse
  - Design patterns
- No lost expressiveness (almost)
- Extendable by users / Common tools
- User base used for feedback (~12+2+36)
- Designed for usability by CS/HCI/Psy/IS team

And now, to the Details
Programming in the Herbal high-level language involves instantiating objects using these ontological classes. Programming a model is reduced to instantiating objects from a set of fixed classes, instead of coding the classes and structure implicitly in a large set of heterogeneous Soar productions.
Herbal IDE Architecture

- Two main responsibilities:
  - make it easy to graphically generate a Herbal high-level program
  - translate the program into Soar productions

Creating a Model

- Working memory
  - The classes that represent the problem domain are created using Protégé, and referenced by other components of the model.
- PSCM
  - States, operators, elaborations, impasses
- Compiling the model
What’s on the CD?

- Protégé 3.0, tutorial, & manual
- Java 1.5 (mac/pc)
- Soar 8.5 & 8.6 (mac/pc)
- Herbal 0.9b & tutorial-manual
- dTank
- Papers (BRIMS, SEIDS, AAAI)
- Slides

Protégé Tutorial

- WORKING IN PAIRS....
- Start up Protégé and use tutorial (~40 min.)
0. If you registered late, contact me (card, note, paper, email) to get the nice summary email http://acs.ist.psu.edu/herbal/

If you have a PC (i.e., not Mac):
1. If you can get on the Internet, please go to http://acs.ist.psu.edu/herbal and locate the Soar 8.5.2 Installer; download and install it.

If you require assistance and/or do not have Internet access, please ask Geoff or Bill for help.

2. break is 230-245

Break Time!

- 10 minutes!
Blocks World Model 1 of 5

- WORKING IN PAIRS!!

- Do Sections 4.1 to 4.3 (10 min.)

Blocks World Model 2 of 5

- WORKING IN PAIRS!!

- Do Sections 4.4 to 4.5 (5-10 min.)
Blocks World Model 3 of 5

- WORKING IN PAIRS!!
- Do Sections 4.6 to 4.7 (5-10 min.)

Blocks World Model 4 of 5

- WORKING IN PAIRS!!
- Do Sections 4.8 to 4.10 (10 min.)
SPECIAL Comments for Section 4.8

- Payoff: how hard is it to change the attribute from “clear” to “movable”
- Issue: graphical method for creating conditions/actions instead of hand coding Soar
- Issue: matching of condition and action pairs because the communicate via the operator memory

Blocks World Model 5 of 5

- Do Sections 5.0, 5.1, and load into Soar (10 min.)
Blocks World with Tie

- Do Section 6 (15 min.)

Herbal: dTank Microworld

acs.ist.psu.edu/dTank

- Java-based game for testing explanation of dynamic, adversarial models
- Multiple players and teams on multiple machines
- Improved 2004: complexity, vision theory, speed, interface, architecture use: 4 (cast, soar, jess, java) + 3 (act-r, jack, CoJACK**)
- Likely to be included in ACT-R and COJACK distributions
- Used by an Army MURI (Sun et al., 2004), at Lock Haven U. (Cohen, 2004), Federal U. of Uberlandia (Brazil)
dTank Model

- WORKING IN PAIRS

- Do Section 7.1-7.3 (~20 min.)

- Pick one of problems in 7.4 to discuss/try

Viewer
Herbal: VISTA Display Designed for Sequence Representation, CaDaDis, integration
acs.ist.psu.edu/CaDaDis, act-r, soar, jess, + jack, cast

Summary

- High level compiler for Soar, designed for other architectures
- Supports model reuse
- Supports explanations
- Example meta-architecture

Future

- More explanations
- More example models
- Slides, handouts, instructions, code, CDs
References (acs.ist.psu.edu/papers/)