Agents in Soar and UM-PRS

Sayan Bhattacharyya
University of Michigan

Object: Exploration of the design space of plan execution architectures

Architectures: Soar, UM-PRS, RAPS,...

Plan execution architectures

Agents: Agents execute plans in dynamic environments in which exogenous events may occur.

Parameter of interest: Sensitivity of plan execution to the conditions of execution

Approach to this work: Experimental/empirical with agents built using existing architectures (as opposed to theoretical)

Salient features of UM-PRS

1) Programmer -initiated i/o:
   "i/o not architectural"
   more than/less than one update per cycle possible

2) Arbitrarily large grain size:
   knowledge areas permit procedural code
   (loops, conditionals, function calls…)

3) No distinction between i-support and o-support
   (everything is equally persistent)

Task and environemnt selection

1. Use same representation for i/o interface consistently.
   (use Soar's input link repesentation)
2. Use hierarchical tasks
3. Use a world which keeps changing fast.

Testbed: Enhanced version of eaters, to support the above.

How architectural differences lead to functional differences

Architectural differences

persistence issues

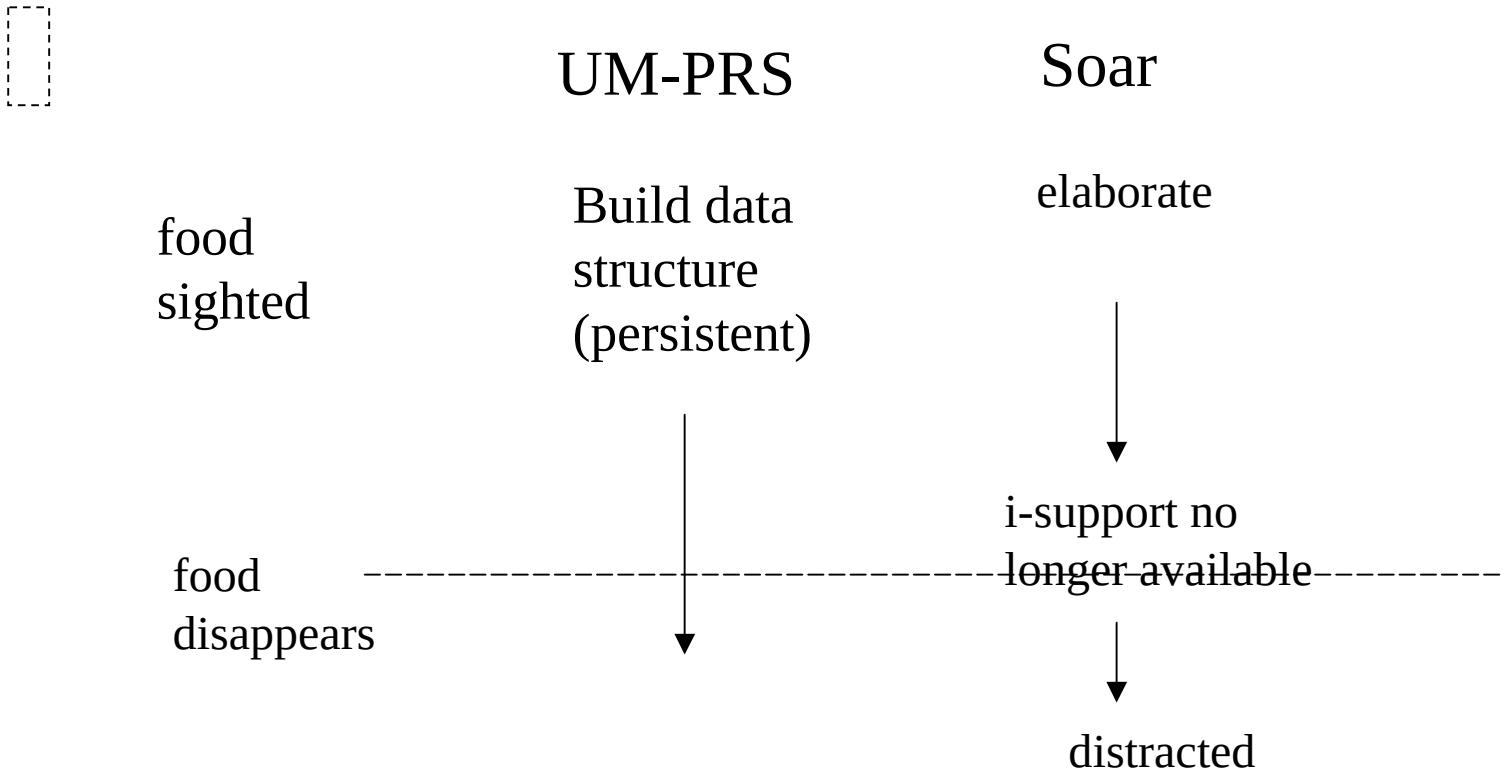
i/o issues

grain size issues

Functional differences

sensitivity to conditions of execution

How architectural differences lead to functional differences (contd)



The agents

S -- a “typical” Soar agent
SU -- a “UM-PRS-like” Soar agent
U -- a “typical” UM-PRS agent
US -- a “Soar-like” UM-PRS agent

SU -- avoids i-supported elaborations in favor of
o-supported data structures
programming trade-off: much more complex
search control
US -- frequent polling of input to mimic sensitivity
of Soar to input changes
programming trade-off: slowed down in
redundant polling

The experiments tell us...

Experiments varying board size, for U,S,SU, US for a “vertical” task hierarchy and “branchy” task hierarchy.

Observations:

1) “Vertical” hierarchy is more well-behaved
2) SU and US data are less regular than S and U data
3) Is sensitivity of plan execution to conditions of execution a curse or a blessing?

   Answer: It depends.

   Will there be a transition point ?

Relation to RAPs

Sensory input part of task methods (like UM-PRS)
But, can support unsolicited sensory information
(like Soar)

Unique names assigned to objects, as long as in sensor range (somewhat like i-support in Soar).

Tasks selected by a RAP can have arbitrarily large grain size (like UM-PRS)

Predictions about RAPs can be made.

Nuggets and coal

Nuggets: Possibly of wider interest than just to Soar and to UM-PRS

Execution is increasingly important as AI gets fielded more in the real world.

Coal Results may be biased by programmer's knowledge and competence levels.