Verification of Decision Procedures Modeled in Intelligent Agents

S. Bhattacharyya+, T. Eskridge+ , M. Carvalho+ and J. Davis *
+ Florida Institute of Technology
* Rockwell Collins, Advanced Tech Center

OVERVIEW

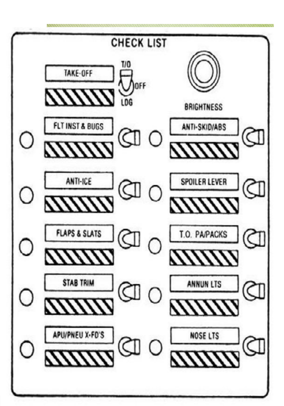
- Desire: Autonomy implements pilot behavior
  - Create verifiable autonomous behaviors that work with human intent to support more cooperative autonomous mission operation
- Approach: Design a cognitive system with formal methods for assurance
  - Design intelligent agent in cognitive framework
  - Translate from cognitive to formal environment
  - Understand assumptions and potential near-term limitations on autonomy
- Objective: Developing trust for intelligent systems

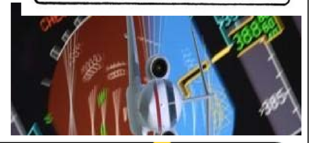
TECHNICAL SUMMARY

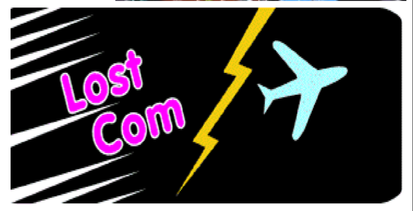
- **Evaluated training manuals to identify requirements for expected pilot behavior**
  - Practical Training Standards
  - ONR
- **Evaluated intelligent learning behavior**
  - Investigated ACT-R (synthetic teammate) and Soar for agent-based behavior modeling
  - Evaluated learning mechanisms
    - Implemented Reinforcement Learning
    - Semantic Memory
- **Developed formal approach to verify composition of rules**
  - Gain trust in autonomy with models in UPPAAL
  - Maintaining architectural integrity
- **Developed translation formalisms from cognitive architectures to formal representation**
  - Maintain architectural integrity
  - Algorithms to translate by maintaining the logic of operations

PILOT MODEL: REQUIREMENTS

- **The system shall be capable of determining whether aircraft systems and equipment are functioning normally**
  - Example: checklists
- **The system shall be capable of recovering from flight plan deviations.**
  - Implemented in UPPAAL
- **The system shall be capable of recovering from unusual attitudes.**
- **The system shall be capable of recovering from lost communications.**







Guarantee that the autonomy always executes the correct behavior as indicated in the FAA standards
- readily implementable
- modular architectural approach

COGNITIVE ARCHITECTURE

- Agent architecture
  - Integration of several components
    - Perception, Memory, Production systems

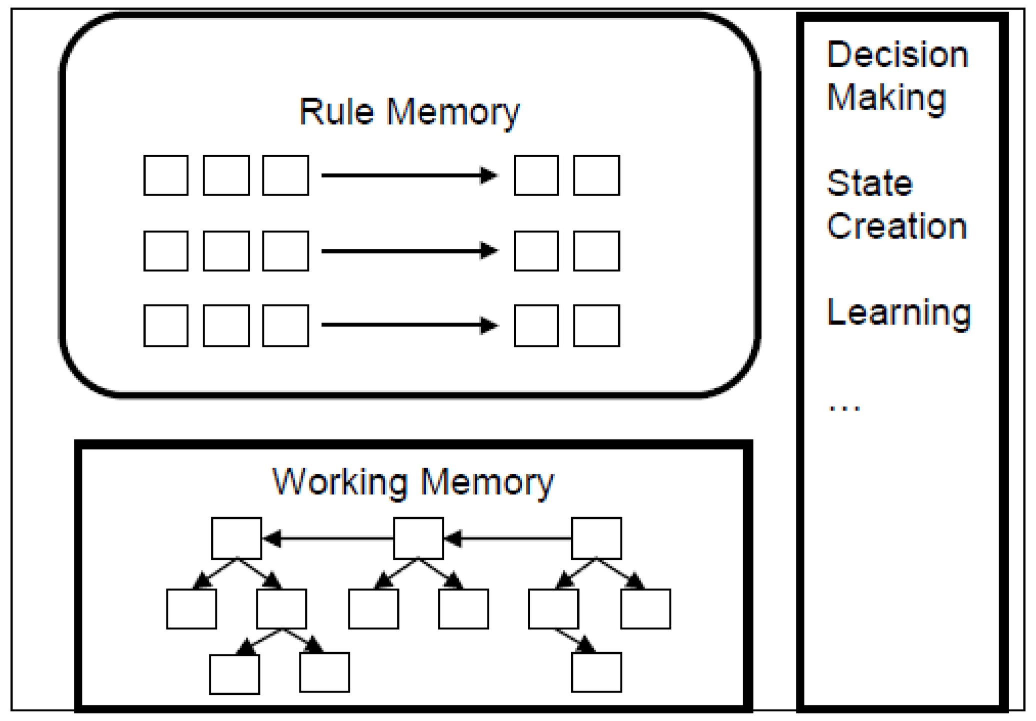
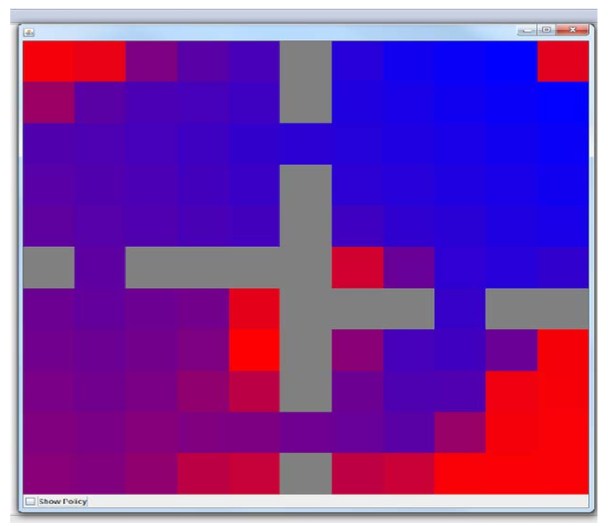


Image Source: Soar Tutorial Part 1

INTELLIGENT LEARNING SYSTEM

- **Applied Reinforcement Learning (RL)**
  - Explores the possible paths to go from source to destination
    - When paths from source to destination in an environment are unknown
  - Generates soar rules based on the path found by the RL

Path Exploration



Paths found

```
0 0 north 1.0 -> 0 1 north 1.0
0 1 north 1.0 -> 0 2 north 1.0
0 2 north 1.0 -> 0 3 north 1.0
0 3 north 1.0 -> 0 4 east 1.0
0 4 east 1.C -> 1 4 north 1.0
1 4 north 1.0  > 1 5 north 1.0
1 5 north 1.0 -> 1 6 east 1.0
1 6 east 1.C -> 2 6 east 1.0
2 6 east 1.C -> 3 6 north 1.0
3 6 north 1.0 -> 3 7 east 1.0
3 7 east 1.C -> 4 7 north 1.0
4 7 north 1.0 -> 4 8 east 1.0
4 8 east 1.C  > 5 8 east 1.0
5 8 east 1.C -> 6 8 east 1.0
6 8 east 1.C -> 7 8 east 1.0
7 8 east 1.C -> 8 8 north 1.0
```

Resultant Ruleset

```
sp {apply*checklist-1
(state <s> ^x 0 ^y 0 ^condition <c> ^operator <o>)
(<c> ^name unchecked)
(<o> ^name checklist-1)
 -->
(write (crlf) |Do checklist item 1|)
(<s> ^operator <o> + )
 (<o> ^name checklist-2)
}

sp {apply*checklist-2
(state <s> ^x 0 ^y 0 ^condition <c> ^operator <o>)
(<c> ^name unchecked)
(<o> ^name checklist-2)
 -->
(write (crlf) |Do checklist item 2|)
(write (crlf) |Done with checklist|)
(<s> ^condition <c> - )
 (<c> ^name checked)
}
```

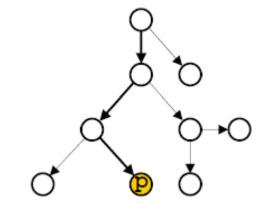
FORMAL VERIFICATION: UPPAAL

- Verification tool: UPPAAL (graphical, supports temporal-logic specification)
- C-based TEJA/UPPAAL converter has been developed
- Uppaal is a Real-time verification tool
- Uppaal consists of three main parts:
  - an editor (description language),
  - a simulator and
  - a model-checker

TEMPORAL CONSTRUCTS IN UPPAAL

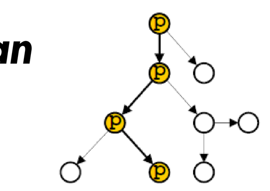
A specific condition ***holds in some state of the*** *model's potential behaviors*

- **E<> p "Exists eventually p"**

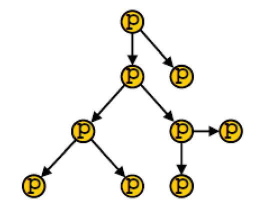


A specific condition ***holds in all the states of an*** *execution path*

- **E[] p "Exists globally p"**

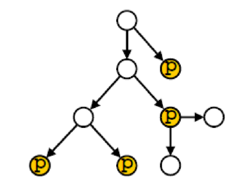


- **A[] p "Always globally p"**

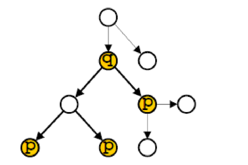


A specific condition ***is guaranteed to hold eventually:***

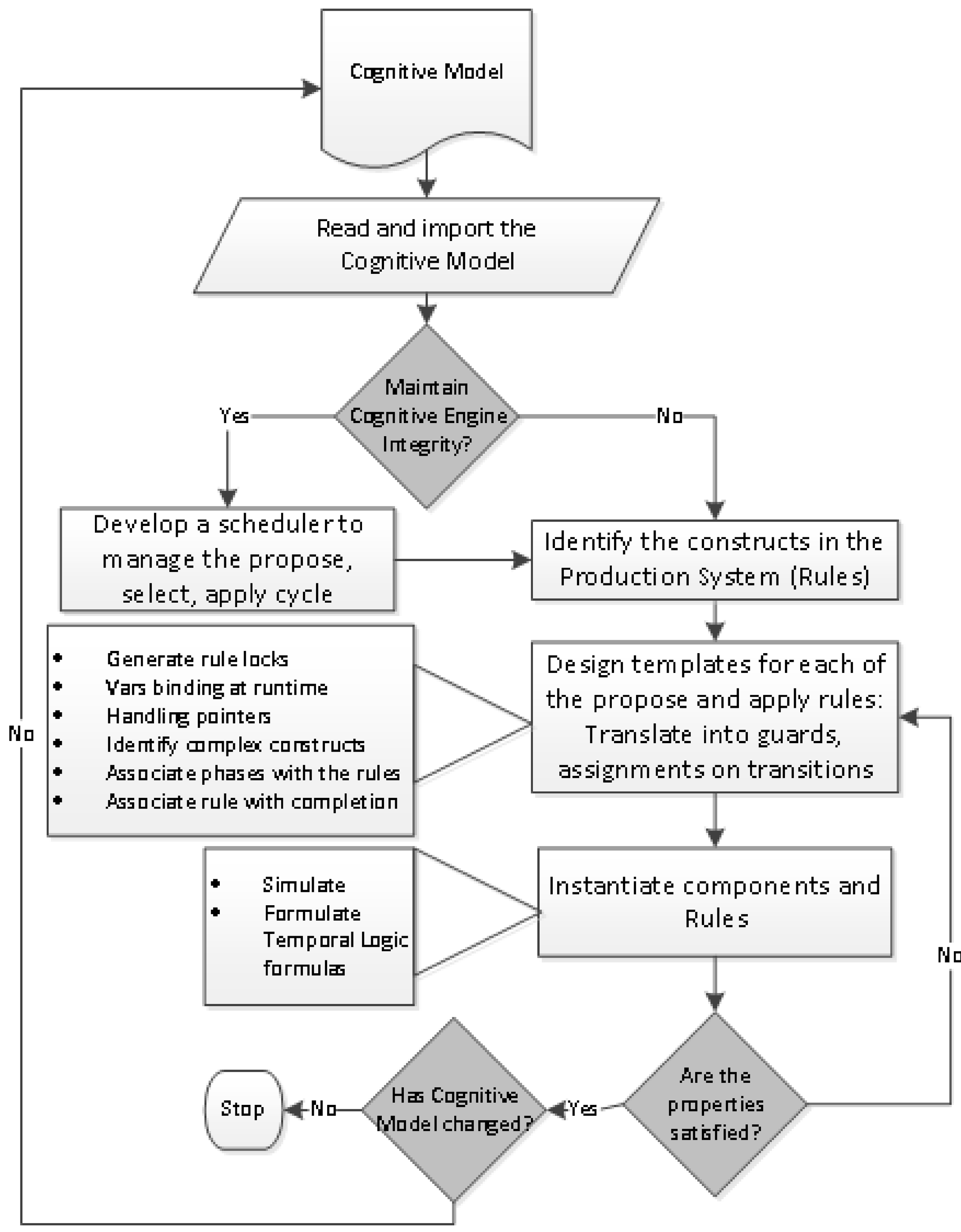
- **A<> P "Always eventually p"**



- **q→ p "q always leads to p"**

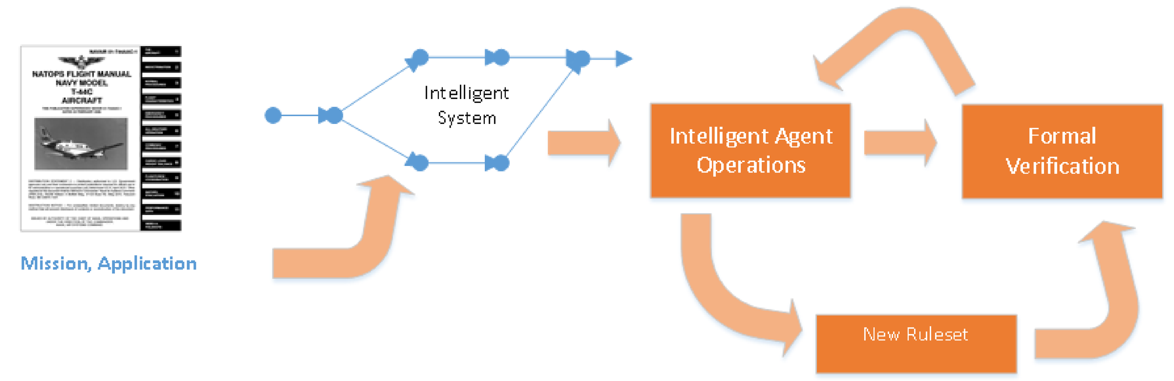


COGNITIVE MODEL TO FORMAL REPRESENTATION

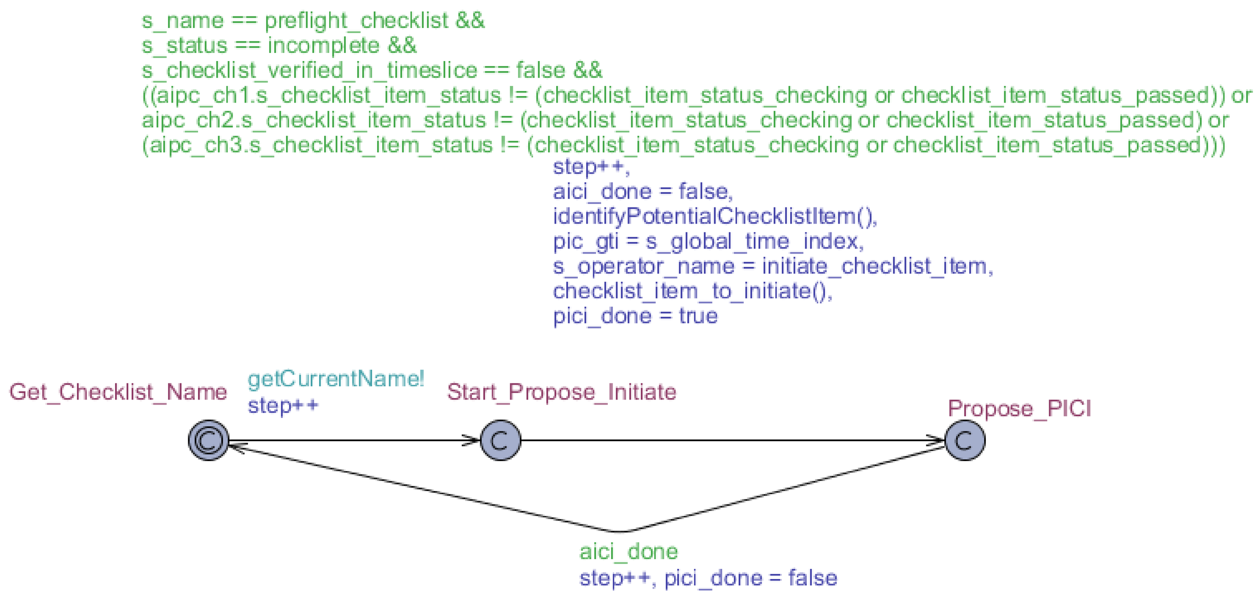


COGNITIVE MODEL – OFFLINE LEARNING WITH V&V FLOW

- Existing checklists are coded into a Soar graph representation
- Soar agent used to learn best path through graph
- New rules formally verified and then used by online Soar agent
- Diagram below illustrates checklist application



MODEL AND PROPERTY SPECIFICATION IN UPPAAL FOR THE PILOT AGENT



Guarantee correctness of design and implementation
Identify conflicts/dependencies in rules
Properties:
- The AS shall start checking items by a certain execution steps
- The autonomous systems (AS) shall check all the listed items
- The AS shall complete the cycle of execution between a range of execution steps

```
A<> aipc_ch3.s_checklist_item_status == checklist_item_status_checking
A<> aipc_chl.s_checklist_item_status == checklist_item_status_passed
A<> step == 5 && pici.Start_PICI && setStepFirstCycle == false
A<> step == 6 && aipc_chl.s_checklist_item_status == checklist_item_status_ready_to_check  && setStepFirstCycle == false
A<> step == 17 && aipc_chl.s_checklist_item_status == checklist_item_status_checking && setStepFirstCycle == false
A<> step == 21 && setStepFirstCycle == false
A<> step == 3 && aipc.Apply_AIPC && setStepFirstCycle == false
```

CONCLUSION

General findings with the research community on Cognitive architecture:
- not rigorously analyzable
  - complex constructs
- not easy to implement

Need to elaborate approach towards verification of cognitive engine

Verification in Uppaal:
- dealing with bindings at runtime
- addressing the handling of pointers
- need for a pre-operator

Florida Institute of Technology
Harris Institute for Assured Information
Rockwell Collins