Incremental Reasoning Agent

Matthias Loebach
1 Problem Statement

2 Knowledge Base & Inference Engine

3 CLIPS

4 World Model

5 Reaction and Interaction with the World

6 Summary

7 Hands-On
Overview

1 Problem Statement

2 Knowledge Base & Inference Engine

3 CLIPS

4 World Model

5 Reaction and Interaction with the World

6 Summary

7 Hands-On
Problem Statement

The robot has a strategic mission (maximal score)
The mission can only be achieved with the right decisions

- High complexity and dynamic of production chains
- Incomplete knowledge
- Plan breaks due to non-deterministic events
- Communication breakdowns
- Robot breakdowns
Why an Expert System?

World State Representation Through Knowledge Base

- Readability:
  - Readable representation of the current world state
  - Situation specific rules
  - Rules are non-scoped (no if-else constructs)

- Transparency: reasoning is easily understandable

- Adaptability: simple extension of the world model
Behavioral Architecture

Agent

Behavior Engine

- Localization
- Motion
- Vision
- ...

Deliberation
Decision making/planning

Reactive Behaviors
Skill execution/monitoring

Components
Actuators/Sensors
Data Processing
Overview

1. Problem Statement

2. Knowledge Base & Inference Engine

3. CLIPS

4. World Model

5. Reaction and Interaction with the World

6. Summary

7. Hands-On
Knowledge Base & Inference Engine

Knowledge Base

- run-time
- external
- sensory
Knowledge Base & Inference Engine

Knowledge Base
- run-time
- external
- sensory

Inference Engine
- rule set
Knowledge Base & Inference Engine

External

Knowledge Base
- run-time
- external
- sensory

Inference Engine
- rule set
Knowledge Base & Inference Engine

- **External**
  - sensors
  - actuators
  - skill-results

- **Knowledge Base**
  - run-time
  - external
  - sensory

- **Inference Engine**
  - rule set
Scope of the Knowledge Base

(co-operative) agent information

- Machine location and type
- Locked positions
Scope of the Knowledge Base

(co-operative) agent information
- Machine location and type
- Locked positions

Game information
- Orders
- Machine state
- Game time
Scope of the Knowledge Base

(co-operative) agent information
- Machine location and type
- Locked positions

Game information
- Orders
- Machine state
- Game time

Sensory information
- Tag detection
- Gripper state
Representing World State in the Knowledge Base

Machine

- Type
- Locks
- Bases
- Position
- State
- Prod Time

Order

- Product-ID
- Complexity
- Gate
- Quantities Ordered
- Quantities Production
- Times
Representing World State in the Knowledge Base

- **Machine**
  - **Type**: Machine type: BS, CS, RS, DS
  - **Locks**: Incoming products, robots
  - **Bases**: In production, at output
  - **Position**: coordinates
  - **State**: IDLE, PRODUCING, etc.
  - **Prod Time**: production time
Representing World State in the Knowledge Base

- **ID from product definition**
- **Product complexity: C0, C1, ...**
- **Delivery gate: 1, 2, 3**
- **Count of ordered/delivered**
- **Count of in production/delivery**
- **Delivery time window in sec**

<table>
<thead>
<tr>
<th>Order</th>
<th>Product-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Complexity</td>
</tr>
<tr>
<td>Gate</td>
<td>Quantities Ordered</td>
</tr>
<tr>
<td>Quantities Production</td>
<td>Times</td>
</tr>
</tbody>
</table>
Overview

1. Problem Statement
2. Knowledge Base & Inference Engine
3. CLIPS
4. World Model
5. Reaction and Interaction with the World
6. Summary
7. Hands-On
Rule-based Production Systems

- First-Order Logic forward chaining systems
- Productions: condition-action rules
- Working memory holds facts (“short-term memory”)
- Rules encode heuristic knowledge (“long-term memory”)
CLIPS

C Language Integrated Production System – CLIPS

- Graph-based Rete-Algorithm
- Lisp-style syntax
- Typically large rule bodies and relatively small number of facts
  - Number of rules: 224
  - Number of fact templates: 49
  - Number of fact changes: >200000 in 10 minutes
- Integrates nicely with C/C++
CLIPS Terminology

**Facts**  Information in Working Memory

(machine (name M-CS1) (mtype CS) ...)

- Incomplete Knowledge
  - Explicitly, e.g. (mtype UNKNOWN)
  - Non-existence of facts
CLIPS Terminology

**Facts**  Information in Working Memory

(machine (name M-CS1) (mtype CS) ...)

- Incomplete Knowledge
  - Explicitly, e.g. (mtype UNKNOWN)
  - Non-existence of facts

**Functions**  Procedural Knowledge

(deffunction get-output (?mps)
  "Return the navgraph point of the output side of the given mps"
  (return (str-cat ?mps "-O")))
Rules  Heuristic Knowledge

(defrule rule-name
  ?m <- (machine (name C-DS))
  =>
  (modify ?m (mtype DS)))
CLIPS Terminology

**Rules**  Heuristic Knowledge

```
(defrule rule-name
    ?m <- (machine (name C-DS))
=>
    (modify ?m (mtype DS))
)
```

**Agenda**  Currently active rules

- List of all rules that have their conditions met
- Rules with higher salience are placed higher on the agenda
- Only one rule active (executed) at a time
Overview

1. Problem Statement
2. Knowledge Base & Inference Engine
3. CLIPS
4. World Model
5. Reaction and Interaction with the World
6. Summary
7. Hands-On
World Model - Overview

Ordered Facts

(game-time 217 771447.0)
(team-color CYAN)
(holding NONE)
(phase PRODUCTION)
(state IDLE)

Structured Facts

(ring (color BLUE) (req-bases 1))
Simple Machine Fact

(machine (name C-BS) (team CYAN) (mtype BS) (incoming) (incoming-agent) (loaded-id 0) (produced-id 0) (state IDLE))
World Model - Machines

Simple Machine Fact

(machine (name C-BS) (team CYAN) (mtype BS)
 (incoming) (incoming-agent) (loaded-id 0)
 (produced-id 0) (state IDLE))

Cap Station Fact

(machine (name M-CS2) (team MAGENTA) (mtype CS)
 (incoming) (incoming-agent) (loaded-id 0)
 (produced-id 611270821) (state READY-AT-OUTPUT))
(cap-station (name C-CS2) (cap-loaded NONE)
 (assigned-cap-color BLACK))
Order Fact

(order (id 1) (product-id 821171164)
  (complexity C0) (delivery-gate 3)
  (quantity-requested 1) (quantity-delivered 0)
  (begin 0) (end 900)
  (in-production 0) (in-delivery 0))
Product Order Fact

(product (id 821171164) (product-id 0) (rings) (cap BLACK) (base RED))

Real Product Fact

(product (id 611270821) (product-id 821171164) (rings) (cap NONE) (base RED))
Overview

1. Problem Statement
2. Knowledge Base & Inference Engine
3. CLIPS
4. World Model
5. Reaction and Interaction with the World
6. Summary
7. Hands-On
Rule Components

Formulate rules to

- Process information
- Trigger skills
- Communicate with other agents
- Check world model sanity
Rule Components

Formulate rules to

- Process information
- Trigger skills
- Communicate with other agents
- Check world model sanity

Rule Syntax

(\textbf{defrule} <rule-name> [\textit{<comment>}]  
  [<textit{declaration}>] ; Rule Properties  
  \textit{<conditional-element>\ast } ; \textit{Left-Hand Side (LHS)}  
  =>  
  \textit{<action>\ast } ; \textit{Right-Hand Side (RHS)}  
)
Binding of Variables

Bind variables in rules to

- Match facts against each other
- Modify fact
- Re-use their value
Binding of Variables

Bind variables in rules to

- Match facts against each other
- Modify fact
- Re-use their value

Binding and matching variables

(defrule example
  (team-color ?team)
  ?m <- (machine (name C-RS2) (team ?team)
    (loaded-id ?id&~NULL))
=>
  <action>
)
Which knowledge has to be added to the fact base?

- New task to start a production step
- Correction for broken world model
- Locks from other agents
Which knowledge has to be added to the fact base?

- New task to start a production step
- Correction for broken world model
- Locks from other agents

Assert New Fact

(\texttt{assert (task (state proposed))})

...
Why remove old facts?

- Obsolete facts clutter the knowledge base (e.g. old steps)
- Facts may represent an old world state (e.g. a base in the gripper)
- Rules might fire continuously due to unchanged conditions
Changing the World – Cleaning Up

Why remove old facts?

- Obsolete facts clutter the knowledge base (e.g. old steps)
- Facts may represent an old world state (e.g. a base in the gripper)
- Rules might fire continuously due to unchanged conditions

Retract a Fact

The fact is specified via the bound fact address.

(retract ?step)
Changing the World – Modifying Facts

Existing facts have to be constantly updated with knowledge

- Most new knowledge is only partial
- New knowledge mostly changes existing facts
- Facts can be selectively modified instead of re-writing
- Increases readability and maintainability
- Executes a retract and an assert
- No modify for *ordered facts*!
Existing facts have to be constantly updated with knowledge

- Most new knowledge is only partial
- New knowledge mostly changes existing facts
- Facts can be selectively modified instead of re-writing
- Increases readability and maintainability
- Executes a retract and an assert
- No modify for ordered facts!

Modifying a Fact

The fact is specified via the bound fact address. If the fact is structured only fields being modified have to be listed.

(modify ?task (state running))
The right-hand side may also contain simple conditions

- If the additional conditional element is only a single value
- If additional facts have to be asserted in some cases
- Avoids clutter with a lot of similar rules
The right-hand side may also contain simple conditions

- If the additional conditional element is only a single value
- If additional facts have to be asserted in some cases
- Avoids clutter with a lot of similar rules

Conditional Change

```lisp
(if (not ?base) then
    (retract ?hf)
    (assert (holding NONE))
    (printout error "'Lost base during drive_to'")
crlf)
```
(defrule discard-unneeded-base
  (declare (salience ?*DISCARD-UNKNOWN*))
  (phase PRODUCTION)
  (state IDLE)
  (team-color ?team-color &~ nil)
  (holding ?product-id &~ NONE)
  (product (id ?product-id))
  (machine (mtype RS) (name ?rs) (team ?team-color))
  =>
  (bind ?task-id (random-id))
  (assert
   (task (name discard-unknown)
     (id ?task-id) (state proposed)
     (priority ?*DISCARD-UNKNOWN*)
     (steps (create$ (+ ?task-id 1))))
   (step (name discard) (id (+ ?task-id 1))
     (task-priority ?*DISCARD-UNKNOWN*)))
Overview

1. Problem Statement
2. Knowledge Base & Inference Engine
3. CLIPS
4. World Model
5. Reaction and Interaction with the World
6. Summary
7. Hands-On
Utilize the agent to make strategic decisions in the game

- Represent your knowledge in the fact base (knowledge base)
- Make decisions on the basis of the current world state
- Use the skiller to interact with the world
- Local distributed incremental agent
- Biggest opportunity for diversification in the RCLL
Overview

1 Problem Statement

2 Knowledge Base & Inference Engine

3 CLIPS

4 World Model

5 Reaction and Interaction with the World

6 Summary

7 Hands-On
First Task

- Recognize the start of production phase
- Drive via way-points on a navigational graph

Afterwards you should be able to:

- React to game phase changes
- Call skills
- Handle finished skills
- Model a production process through consecutive skill calls
Second Task

- Collect base elements from the base station
- Process the base element in a cap station
- Deliver the product to the delivery station

Afterwards you should be able to:

- Pickup and insert base elements into machines
- Communicate with production machines
- Assemble a product of C0 complexity
Agent source code is located in

`~/fawkes-robotino/src/agents/ws15/`

**CLIPS** agent webview

http://localhost:8081/clips/agent

**CLIPS** Programming Guide


Start the simulation with this command

```
./gazsim.bash -x start -r -n1 -a
```