Non-Monotonic Reasoning for Localization in RoboCup

David Billington
Vladimir Estivill-Castro
Rene Hexel
Andre Rock

Australia
Outline

- Reasoning and Localization
- Why reasoning and modelling with logic
  - The Software Engineering justification
  - The Hybrid Intelligent Agent justification
- Running reasoning on a AIBO ERS-7
- Model Development and Results
- Conclusion
Reasoning

- Deriving conclusions from facts
  - Apparently, a fundamental characteristic of intelligence
- An expected aspect of intelligent systems
- Withdrawing conclusions in the light of new evidence is a capability usually referred to as non-monotonic reasoning
Our environment

RoboCup

- A test-bed for Multi-Agent Systems

- We know our environment, so one would expect to be able to construct a knowledge base and apply reasoning
RoboCup environment is hard

- Non-deterministic
  - I can not predict the state of the environment after I perform an action
- Not accessible
  - I can not sense all elements of the environment
- Dynamic
  - Environment changes while I decide what action to take
- Teams
  - I need to negotiate, collaborate, distribute tasks and goals
- Adversaries
  - Of unknown capabilities
We demonstrate reasoning within the task of localization

- Dynamically selecting proper inputs for localization
  - The classical example in RoboCup for the Aibo league is that
    - A frame where both goals are visible indicates something wrong with the object recognition task
Possible solutions

- Introduce sanity checks
  - Filter out the frame if both goals are visible
- Pass it out to localization and expect the sophistication of the algorithm (capacity to handle error in sensor input) to handle these cases
  - Kalman Filter
  - Markov Localization
  - Monte-Carlo localization
Our approach

- Vision and Object Recognition
- Consistency Module
- Localization Algorithm
Our approach

Consistency Module

Non-monotonic logic that combines facts known about the environment with what is reported as visible in this frame
Why non-monotonic logic

- To reason about the inconsistent information provided by the sensors (vision)
- Without reasoning, all localization methods must determine
  - $\text{Prob}(\text{visible scene} | \text{position})$
Problem with localization methods

- Illustration
  - \( \text{Prob} \ (\text{front goal visible} \& \text{back goal visible} \mid \text{position}) = 0 \)
    - Not the best answer, or defines a large set of special cases
    - It is hard to express it as function of
      \( \text{Prob} \ (\text{front goal visible} \mid \text{position}) \)
      and
      \( \text{Prob} \ (\text{front goal visible} \mid \text{position}) \)
Plausible logic

- Only non-monotonic logic with an efficient non-looping algorithm
- Can prove using factual information and also plausible information
- 3 types of rules
  - \( A \rightarrow l \) (factual information)
    - \( \text{Human}(x) \rightarrow \text{Mammal}(x) \) [All humans are mammals]
  - \( B \Rightarrow f \) (plausible situations)
    - \( \text{Bird}(x) \Rightarrow \text{Fly}(x) \) [Birds usually fly]
  - \( A \supseteq \neg l \)
    - \( \{\text{sick}(x), \text{Bird}(x)\} \not\Rightarrow \text{Fly}(x) \) [Sick birds may not fly]
Plausible logic (cont)

- Rules are in an acyclic hierarchy
  - $R_i > R_j$
    - Rule $i$ is more informative than rule $j$.
- Conclusion with one rule may be defeated by the more informative rule
Why reasoning and modelling with logic

The Software Engineering justification

- All the "intelligence" (logic) about what makes sense in an image (or sequence of images) is properly encapsulated in a human understandable logic
  - Not a series of "if ..then …else" statements of C++ in the code
  - Can test completeness and correctness
  - Can be updated easily

The Hybrid Intelligent Agent justification

- A higher level description that allows reasoning
Illustration

Naturally to develop rules systems where the new rules redefine exception to the previous ones

- 3 laws
  1. A robot may not harm a human
  2. A robot must obey a human unless it contradict law 1
  3. A robot must protect itself unless contradicts rule 1 or 2

- Ripple down rules
  - Rules are defined and new rules are subsequently added to revise the cases not covered by the more general rules
  - A tree that is a hierarchy of rules
    - No formal reasoning
Modelling with standard logic

- A first model

\{\text{See}(x)\} \cup \{\neg \text{See}(y) : y \in \text{Landmarks}-{x}\} \rightarrow \text{Cs}(x)

- If I only see one object, then it is consistent
Modelling with standard logic

1. A second model

\[ C(x,y) = \{ \text{See}(x), \text{See}(y) \} \cup \{ \neg \text{See}(z) : z \in \text{Landmarks}-\{x,y\} \} \]

1. \( \{ \text{SeeLtoR}(x,y), \text{FactLtoR}(x,y,z) \} \cup C(x,y) \rightarrow Cs(x,y) \)
2. \( \{ \text{SeeLtoR}(y,x), \text{FactLtoR}(x,y,z) \} \cup C(x,y) \rightarrow Cs1(x,y) \)
3. \( \{ Cs1(x,y), \text{Post}(x), \text{Goal}(y) \} \rightarrow Cs(x) \)
4. \( \{ Cs1(x,y), \text{Post}(x), \text{Post}(y), \text{BigSmall}(x,y) \} \rightarrow Cs(x) \)

The world

x y z
Modelling with standard logic

A second model

\[ C(x, y) = \{ \text{See}(x), \text{See}(y) \} \cup \{ \neg \text{See}(z) : z \in \text{Landmarks}-\{x, y\} \} \]

1. \{ \text{SeeLtoR}(x, y), \text{FactLtoR}(x, y, z) \} \cup C(x, y) \rightarrow C_{s}(x, y)
2. \{ \text{SeeLtoR}(y, x), \text{FactLtoR}(x, y, z) \} \cup C(x, y) \rightarrow C_{s1}(x, y)
3. \{ C_{s1}(x, y), \text{Post}(x), \text{Goal}(y) \} \rightarrow C_{s}(x)
4. \{ C_{s1}(x, y), \text{Post}(x), \text{Post}(y), \text{BigSmall}(x, y) \} \rightarrow C_{s}(x)

The world

Vision
Modelling with standard logic

A second model

\[ C(x,y) = \{\text{See}(x), \text{See}(y)\} \cup \{\neg \text{See}(z) : z \in \text{Landmarks}-\{x,y\}\} \]

1. \( \{\text{SeeLtoR}(x,y), \text{FactLtoR}(x,y,z)\} \cup C(x,y) \rightarrow \text{Cs}(x,y) \)
2. \( \{\text{SeeLtoR}(y,x), \text{FactLtoR}(x,y,z)\} \cup C(x,y) \rightarrow \text{Cs1}(x,y) \)
3. \( \{\text{Cs1}(x,y), \text{Post}(x), \text{Goal}(y)\} \rightarrow \text{Cs}(x) \)
4. \( \{\text{Cs1}(x,y), \text{Post}(x), \text{Post}(y), \text{BigSmall}(x,y)\} \rightarrow \text{Cs}(x) \)
Modelling with standard logic

A second model

\[ C(x,y) = \{ \text{See}(x), \text{See}(y) \} \cup \{ \neg \text{See}(z) : z \in \text{Landmarks-}\{x,y\}\} \]

1. \( \{ \text{SeeLtoR}(x,y), \text{FactLtoR}(x,y,z) \} \cup C(x,y) \rightarrow Cs(x,y) \)
2. \( \{ \text{SeeLtoR}(y,x), \text{FactLtoR}(x,y,z) \} \cup C(x,y) \rightarrow Cs1(x,y) \)
3. \( \{ Cs1(x,y), \text{Post}(x), \text{Goal}(y) \} \rightarrow Cs(x) \)
4. \( \{ Cs1(x,y), \text{Post}(x), \text{Post}(y), \text{BigSmall}(x,y) \} \rightarrow Cs(x) \)

The world

Vision
Problems with standard logic

- Rapidly we have the same situation
  - Many different cases coded essentially independently
    - Seeing exactly 3 objects need 26 rules
    - Seeing exactly 4 objects needs 120 rules
  - Proves most C++ is incomplete
    - (and perhaps inconsistent)
    - Survives because of the frame rate
    - Concerns on correctness/reliability of intelligent systems
Implementing Plausible Logic on SONY Aibo

- Besides Plausible Logic we develop a Logic Programming Language - DPL
  - Create definitions, macros, determine what to prove
- A HASKELL implementation of the inference algorithm of plausible logic
  - A program in DPL that proves off-line
    - Finds the equivalent logic expression to Cs(Front goal) in terms of World predicates and Test predicates
- A simulator for validation of-line and gluing code
- A Template method in the consistency module on the Aibo
Model 1

- R1: $\Rightarrow \sim Cs(x)$.
- R2: See(x) $\Rightarrow Cs(x)$.
- R2 $\succ$ R1.

- Validates the system and implementation process
Model 2

- R1: \( \Rightarrow \sim Cs(x) \).
- R2: See\( (x) \Rightarrow Cs(x) \).
- R2\( \Rightarrow \) R1.
- R3: \{See\( (x), See(y), Opp(x, y) \} \Rightarrow \sim Cs(x) \).
- R3\( \Rightarrow \) R2
- R4: \{See\( (x), See(y), SeeLtoR(y, x), LR(x, y) \} \Rightarrow \sim Cs(x) \).
- R4: \{See\( (x), See(y), SeeLtoT(y, x), LR(x, y) \Rightarrow \sim Cs(x) \).
- R4\( \Rightarrow \) R2

© Vladimir Estivill-Castro
Illustration

The left post is correct but the right post and goal are inverted
Model 3

- R1: $\Rightarrow \neg \text{Cs}(x)$.
- R2: $\text{See}(x) \Rightarrow \text{Cs}(x)$.
- R2$\Rightarrow$R1.
- R3: $(\text{See}(x), \text{See}(y), \text{Opp}(x, y)) \Rightarrow \neg \text{Cs}(x)$.
- R3$\Rightarrow$R2
- R4: $(\text{See}(x), \text{See}(y), \text{SeeLtoR}(y, x), \text{LR}(x, y)) \Rightarrow \neg \text{Cs}(x)$
- R4: $(\text{See}(x), \text{See}(y), \text{SeeLtoT}(y, x), \text{LR}(x, y)) \Rightarrow \neg \text{Cs}(y)$
- R4$\Rightarrow$R2
- R5: $(\text{See}(x), \text{See}(y), \text{See}(z), \text{SeeLtoR}(y, z), \text{SeeLtoR}(z, x), \text{Adj}(x, y, z)) \Rightarrow \text{Cs}(y)$
- R5: $(\text{See}(x), \text{See}(y), \text{See}(z), \text{SeeLtoR}(y, z), \text{SeeLtoR}(z, x), \text{Adj}(x, y, z)) \Rightarrow \text{Cs}(x)$
- R5$\Rightarrow$R4
- R6: $(\text{See}(x) < \text{see}(y), \text{See}(z), \text{SeeLtoR}(x, z), \text{SeeLtoR}(z, y), \text{LR}(x, y), \text{LR}(y, z), \text{Opp}(x, z)) \Rightarrow \text{Cs}(x)$
- R6: $(\text{See}(x), \text{See}(y), \text{See}(z), \text{SeeLtoR}(x, z), \text{SeeLtoR}(z, y), \text{LR}(x, y), \text{LR}(y, z), \text{Opp}(x, z)) \Rightarrow \text{Cs}(y)$
- R6$\Rightarrow$R3
- R6$\Rightarrow$R4
Illustration

The left post and goal appear in the correct order, but the right post appears leftmost.
The module in action
Discussion

- CPU times were very positive
  - Model 1: 44 microseconds
  - Model 2: 60 microseconds
  - Model 3: 110 microseconds
    - On ERS-7 SONY Aibo
Conclusion

- The initial progress on logic and reasoning within AI has largely been discarded from mobile robotics in favour of reactive architectures.
- We demonstrate the use of non-monotonic reasoning in the challenging application of RoboCup.
- Plausible logic is the only non-monotonic logic with an algorithm that detects loops.
THANK YOU