Automated Norm Synthesis in an Agent-based Planning Environment

George Christelis

First Supervisor: Michael Rovatsos
Second Supervisor: Ron Petrick

November 13, 2008
Introduction

• Agent actions can conflict in shared systems
• Coordination in a mechanism to resolve these conflicts
• **Social norms** are one form of pre-planning coordination
  • Based on constraining agent behaviour
  • Introduce predictability and improve system efficiency
• This proposal is concerned with the automated design of social norms in planning-based domains
Overview

- Social norms and norm synthesis overview
- Planning background
- Synthesis research proposal and hypothesis
- Work to date: restricted domain synthesis
- Work plan and milestone details
- Questions and comments
Social norms are a set of established, expected patterns of behaviour

- All agents in the system are governed by the same set of norms
- Agents should behave in a way that is consistent with the norms

Norms allow us to define what behaviour is socially acceptable in the system.
Social Norms

Key properties of social norms:

- **Explicit Representation** - adoption, adaptive, dynamic, reasoning
- **Persistent** - long term
- **Generality** - generally applicable for multiple states, open-textured
- **Incentives / Deterrents** - sanctions, rewards

Norms are implemented via systems of regimentation or enforcement.
Benefits of Social Norms

- Means to achieve a social objective
- Introduces system predictability
- Reduce agent reasoning, opponent modelling
- Less conflict resolution, less communication
- Persistence, no renegotiation
Problem: The Synthesis of Norms

- Consider an abstract state transition system.
- Classify states as conflict or conflict-free states.
- Consider prohibitionary norms that forbid actions from certain states.

How can we create norms that avoid conflict states, while ensuring access to conflict-free states? Can it be automated?
Related Work

1. Naive Approach
   - State enumeration required
   - Many resulting norms, each conditional on a unique state

2. Artificial Social Systems
   - NP-complete in general
   - No synthesis process provided

3. Robot Mobilisation
   - Polynomial synthesis
   - Very restricted problem

4. Social Laws in Alternating Time
   - Joint state enumeration required
Improvement is Possible

In the general case synthesis is intractable!

In practice we can take advantage of:

• declarative domain specification
• state abstraction techniques
• knowledge of agent goals
• online heuristic-based synthesis techniques
• offline computational resources
We can automated the synthesis of prohibitionary norms given a declarative domain description a conflict specification to efficiently avoid conflict states so that:

1. a complete joint state enumeration is not always necessary, and
2. sanctions can be synthesised for enforcement-based systems, and
3. the process is anytime, resulting in improved synthesised norms over time.
Planning Based Norm Synthesis

We adopt a number of concepts from existing planning work:

- Declarative domain and operator specifications
- State specifications abstract over system states
- Existing planning-based search heuristics
General Propositional Planning Formalism

Some planning background:

- $\hat{\Sigma}$ the literals over the propositional atoms $\Sigma$
- $\mathcal{L}$, the language of propositional logic over $\Sigma (\land, \lor, \neg)$
- A state $s$ is a **complete** truth assignment over $\hat{\Sigma}$
- Set of operators $O$, each of the form $\langle pre, post \rangle$
- State specification transition function $R : 2^{\hat{\Sigma}} \times O \rightarrow 2^{\hat{\Sigma}}$
State specifications model groups of states:

- a state specification $S$ is a partial truth assignment over $\hat{\Sigma}$
- consistency and completeness
- A run is a state specification/operator sequence of the form $S_0 \xrightarrow{o_1} S_1 \xrightarrow{o_2} \ldots$

**Example:** If $\Sigma = \{a, b, c\}$, $s_1 = \{b, c\}$ and $s_2 = \{b\}$ are valid states.

If $S = \{\neg a, b\}$, then $S$ models both $s_1$ and $s_2$. 
Planning Restrictions

A number of restricted formalisms exist:

- $S$ - complete (STRIPS)
- $S_I$ - incomplete
- $S_{IC}$ - incomplete, conditional effects
- $S_{BIC}$ - boolean, incomplete, conditional effects

The computational complexity of norm synthesis is dictated by the formalisms chosen.
Work To Date

• Prohibitionary norm representation
  • \( n = \langle pre, op \rangle \) where \((pre \in 2^L, op \in O)\)
  • If a specification \( S \models pre \) then the operator \( op \) is forbidden in \( S \).

• Normative extension to General Planning Problem
  • Plan execution under prohibitions

• Conflict Rooted Synthesis (CRS) process for synthesis in the restrictive \( S_I \) formalism.
We assume:

- incomplete state specifications
- a declarative planning domain specification ($\Xi$)
- a specification of undesirable conflict states, ($S_C$)

Norm synthesis is concerned with avoiding all states modelled by $S_C$, while ensuring accessibility to all conflict-free states.
Conflict-rooted synthesis has three main processes:

- **Traversal**: find all unique runs starting and ending in conflict-free specifications, traversing through conflict specifications.
- **Synthesis**: synthesise norms for each run
- **Accessibility**: check accessibility between each precursor specification, and all successor specifications.
1) We begin with the conflict specification
2) Infer all possible precursor specifications
3) Refine specifications under new constraints
Synthesis in $S_I$

4) Infer all possible successor runs
5) Synthesis Norms
Synthesis in $S_I$

6) Check accessibility
Example: Tunnels Domain

- Two agents, 1 and 2
- Agents can move, enter, exit, idle
- Asynchronous actions
- Goals are to traverse the system
- Collisions are very bad!
- We define $S_C = \{at_1(N), at_2(N)\}$. 
Example: Traversal For Action exit

\[ S_C \]
Example: Traversal For Action $exit_1$
Example: Traversal For Action exit

- exit
- move
- idle
- enter

Node $S_C$ with transitions:
- $exit_1$ to idle
- idle to enter
- idle to move

Nodes:
- $exit_1$ (black)
- idle (red)
- move (blue)
- enter (blue)
Example: Traversal For Action exit

George Christelis
Automated Norm Synthesis in an Agent-based Planning Environment
November 13, 2008
Example: Traversal For Action exit

\[ \ldots \text{at}_1(X), \text{tunnel}(X) \ldots \]
Example: Traversal For Action exit

\[ S_C \]

- exit\(_1\)
- idle\(_2\)
- move\(_2\)
- enter\(_2\)
- enter\(_1\)
- idle\(_1\)
- move\(_1\)
Example: Traversal For Action exit

George Christelis
Automated Norm Synthesis in an Agent-based Planning Environment
November 13, 2008
Example: Synthesised Norms

- We prohibit action $\text{enter}_1(L, T1)$ under the condition:

$$
\{ \text{at}_1(L), \text{at}_2(T2), \text{tunnel}(T2), \text{tunnel}(T1), \neg \text{tunnel}(N), \neg \text{tunnel}(L), \text{conn}(T2,N), \text{conn}(T1,N), \text{conn}(L,T1) \}
$$

That is, an agent should not choose to enter a tunnel, if the exit of the tunnel and the exit of another tunnel in which the other agent is are common.
Limitations of CRS in $S_I$

The work to date has a number of limitations:

- Variable support in CRS?
- Planning between incomplete specifications?
- Less restrictive planning formalisms ($S_{IC}, S_{BIC}$)?
- Sanction Synthesis?
- Norm representation: Obligations?
- Heuristics to tackle complexity?
Work Plan: Incremental Complexity

• Iterative research, implement, evaluate approach
• Start simple, progress to more complex
  • $S_I$, prohibitionary norms, no sanction synthesis
• Three avenues to investigate:
  • Increase planning representation ($S_I \rightarrow S_{BIC}$)
  • Extends synthesis for sanctions and rewards
  • Improve complexity for online synthesis
Work Plan: Overview

- Four iterations, meeting three key objectives:
  - Synthesis in different formalisms
  - Sanction Synthesis
  - Heuristics / Domain Assumptions for Online Synthesis
- Analytic evaluation of algorithms: termination, correctness, completeness (unlikely)
- Three empirical evaluation domains
  - Parcel World, Tunnels, Healthcare
  - Ordering over domains exists
- Iterations are documented upon completion
Empirical Evaluation Criteria

- **Efficiency**
  - empirical evaluation of synthesis
  - comparison with complete joint state enumeration
  - comparison between formalisms

- **Brevity / Generality**
  - norms synthesised per social objective
  - comparison with complete joint state enumeration
  - comparison between formalisms

- **Domain Performance**
  - Brute force norms
  - CRS Synthesised norms
  - Heuristic CRS Synthesised norms
Task and Milestones

Task A: Synthesis for $S_I$ planning formalisms
- 11/2008 → 01/2009

Task B: Extend Synthesis for $S_{IC}$ and $S_{BIC}$
- 02/2009 → 06/2009

Task C: Sanction Synthesis for Enforced Systems

Task D: Heuristics for Online Bounded Synthesis
- 11/2009 → 03/2010
# Work Plan: Gantt Chart

<table>
<thead>
<tr>
<th></th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Impl.</td>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Eval.</td>
<td>3</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>Doc.</td>
<td>7</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

A: $S_I$ B: $S_{BIC}$ C: Sanctions D: Online
Risks and Contingencies

- All tasks are dependent on Task A:
  - Support variables in conflict specification?
  - Extend planning formalism for planning over incomplete specifications?
  - Make assumptions of domain to simplify accessibility.

- Tasks C and D extend Task B
  - Boolean formulae result in complex expressions?
  - OBDD to simplify and generalise expressions?
  - Make assumptions of form of expressions?
  - Tasks C and D can extend Task A instead.
Conclusions

• Social norms are a form of pre-planning coordination
• The synthesis of norms can be improved, given some additional knowledge
• Work to date including the conflict rooted synthesis process
• Outlined future work and milestones
Questions?