Planning with Global Constraints for Computing Infrastructure Reconfiguration

Herry and Paul Anderson

h.herry@sms.ed.ac.uk
http://goo.gl/c4wK3

AAAI-12 Workshop on Problem Solving using Classical Planners
Toronto, July 2012
Outline

• System Configuration
• Current Approaches
• Global constraints
• Proposed Approach
• Example Use-Case: Cloud-Burst
• Conclusion
System Configuration

• Transform computing resources into a functioning system according to a given specification
• Resources – operating system, software, user-account, services, network, etc
• Challenges
  – Size
  – Multi-platform
  – Dependencies
  – Multi-aspect
  – Multi-user
## Current Approaches (Popular)

<table>
<thead>
<tr>
<th>Method</th>
<th>Declarative</th>
<th>Fixed Workflow</th>
<th>Dynamic Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Define goal state</td>
<td>Define workflow</td>
<td>Define goal state</td>
</tr>
<tr>
<td>Tools</td>
<td>Select &amp; execute actions</td>
<td>Execute the given workflow</td>
<td>Generate &amp; execute workflow</td>
</tr>
<tr>
<td>Ordering</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Examples</td>
<td>Puppet, LCFG, BCFG, SmartFrog, Chef</td>
<td>IBM Tivoli, Microsoft Opalis, ControlTier</td>
<td>Keller et.al. ('04), El Maghraoui et.al. ('06), Hagen et.al. ('10), Herry et.al. ('11)</td>
</tr>
</tbody>
</table>
Global Constraints

• Constraints that must be satisfied in the intermediate (during reconfiguration) and goal states
  – DNS service must be available all the time

• None supports global constraints

• User could modify particular actions
  – It's not feasible in practice
Why need Global Constraints?

• Modifying actions is as hard as planning
• Administrator usually does not have deep knowledge of the resources
• Lack of permission
  – Copyright, License
• Separation of concerns
  – Reusability of components
Proposed Approach

- SFplanner
  - Allowing user to define global constraints
  - Use planner to generate workflow
- Modelling the configuration problem
  - SFp language
- Planner
  - Fast-Downward
SFp language

- Extension of SmartFrog (SF) language
- Current
  - Object-oriented
  - Modules
  - Goal & Global constraints
  - Reference
- Future
  - Abstract Data-Types (Array, Set)
  - Create/delete objects
  - Composite

```server.sfp
class Server {
    running false
    action start {
        precondition { }
        postcondition {
            $this.running true
        }
    }
    action stop {
        precondition { }
        postcondition {
            $this.running false
        }
    }
}
```

```client.sfp
class Client {
    refer as *Server
    action changeReference(s as *Server) {
        precondition { }
        postcondition {
            $this.refer $s
        }
    }
}
```
SFp language

desktops.sfp
# include file("client.sfp")
pc1 as Client {
    refer $web1
}
pc2 as Client {
    refer $web1
}
pc3 as Client {
    refer $web1
}

web-servers.sfp
# include file("server.sfp")
web1 as Server {
    running true
}
web2 as Server {
    running false
}

Include file("web-servers.sfp")
#include file("desktops.sfp")

class system1.sfp
constraint goal {
    $web1.running false
}
constraint global {
    $pc1.refer.running true
    $pc2.refer.running true
    $pc3.refer.running true
}
SFplanner

1. Dashboard
2. Goal and Global Constraints
3. Resource Database
4. Planner
5. Current State
6. Execution Manager
7. Workflow
8. Monitoring Manager
9. Execution Agent
10. Monitoring Agent

Managed System
Planning Process

1. Normalisation
2. Compilation
3. Fast-Downward
4. Postprocessing

- SFp Configuration task
- Normalised SFp Configuration task
- FDR Task
- Final Workflow
- Raw Workflow
Example: Cloud-Burst

Initial

Goal
Generated Workflow

• FF Heuristic
Conclusion & Future Works

• SFplanner generates workflow to attain the goal state and also preserving global constraints
• Classical planner could solve configuration problems
• Ongoing/future works
  – Soft global constraint
  – Larger size of use-cases (still collecting)
  – More distributed and localized approach to increase resilient

Source codes: http://github.com/herry13
Thank you

Q & A