Web Service Composition as a Planning Task

Experiments Using Knowledge-Based Planning

Erick Martínez & Yves Lespérance

Department of Computer Science
York University
Toronto, Canada
Motivation

- Next generation Web Services
  - Semantic Web
  - Reasoning / Planning
  - Automation

- Agent-oriented toolkit for advanced MAS / WSC and provisioning

- Previous results (knowledge-based planning)
Automated Web Service Composition as Planning

- **WSC Problem**: given a set of Web services and some user defined task or goal to be achieved, automatically find a composition of the available services to accomplish the task.

- **WSC as a Planning Problem**: 
  - Predefined available services as the building blocks of a plan
  - Many WS actions involve sensing
  - Large search space, incomplete information in the initial state

- **Planner that can generate conditional plans & supports sensing actions**
PKS
(Petrick & Bacchus, 2003)

- Generalization of STRIPS \[\text{[Fikes & Nilsson, 1971]}\]
  - Four Dbs: $K_F$, $K_W$, $K_V$, $K_X$
  - Actions:

<table>
<thead>
<tr>
<th>Action</th>
<th>Precondition</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkFlightSpace(n,d)</td>
<td>$K(\exists\text{Flight}(n,d))$</td>
<td>$\text{add}(Kw, \text{availFlight}(n,d))$</td>
</tr>
</tbody>
</table>

- DSURs: $K(\exists\text{Flight}(n,d)) \Rightarrow \text{add}(Kv, \text{flightNum}(n,d))$
- Goal
- Planning problem: $< I, A, U, G >$
WSC in PKS

- PKS primitive actions correspond to WS
  - Knowledge-producing actions ↔ information gathering WS
  - Physical actions ↔ world-altering WS

- New WS becomes available → add new primitive action to domain specification
WS Representation in PKS

- For each action $A_i$:
  - encode user preferences / customization constraints using $desA_i$ fluent \[McIlraith & Son, 2002\]
  - encode domain specific search control constraints using $indA_i$ fluent

- Generic domain specification
  - action specification (WS)

- Problem specification
  - goal + DSURs (addresses PKS limited expressiveness)
PKS Spec. Example (Air Travel Domain)

| Action               | Precondition                        | Effects                                                              |
|----------------------|-------------------------------------|                                                                     |
| findRFlight(x)       | K(airCo(x))                         | add(Kw, flightExists(x))                                            |
|                      | K(indFindRFlight(x))                | add(Kf, ¬indFindRFlight(x))                                         |
|                      | K(desFindRFlight(x))                |                                                                     |
| bookFlight(x)        | K(airCo(x))                         | add(Kf, bookedFlight(x))                                           |
|                      | K(availFlight(x))                   | del(Kf, availFlight(x))                                            |
|                      | K(indBookFlight(x))                 | add(Kf, ¬indBookFlight(x))                                         |
|                      | K(desBookFlight(x))                 |                                                                     |

Domain specific update rules (DSUR)

\[ K(airCo(x)) \land \neg Kv(flightNum(x)) \land K(flightExists(x)) \Rightarrow add(Kv, flightNum(x)) \]

1 explicit parameter: company
PKS Goal Example
(Prob. BMxF)

- Goal: book a flight with a price not greater than the user's maximum price

\[
\begin{align*}
&\text{/* book company within budget */} \\
&\exists_K (x) [K(\text{airCo}(x)) \land K(\text{bookedFlight}(x)) \land \\
&\quad K(\neg \text{priceGtMax}(x))] \\
&\text{/* no flight booked */} \\
&\text{KnowNoBudgetFlight} \\
&\text{KnowNoAvailFlight} \\
&\text{KnowNoFlightExists}
\end{align*}
\]
User Pref. / Customization
Constraints Example (Prob. BMxF)

- **DSUR 1:**
  \[
  K(\text{airCo}(x)) \land \neg Kw(\text{priceGtMax}(x)) \land \\
  Kv(\text{userMaxPrice}) \land Kv(\text{flightCost}(x))
  \Rightarrow add(Kw, \text{priceGtMax}(x))
  \]

- **DSUR 2:**
  \[
  K(\text{airCo}(x)) \land K(\neg \text{priceGtMax}(x)) \land \\
  \neg Kw(\text{desBookFlight}(x))
  \Rightarrow add(Kf, \text{desBookFlight}(x))
  \]
Experiments - Problem Set

- Set of 5 problems (air travel domain):
  - hard constraints
    - **BPF**: book preferred company, otherwise book any flight
    - **BMxF**: book any flight within budget
    - **BPMxF**: book flight within budget, favour preferred company
  - optimization constraints
    - **BBF**: book cheapest flight
    - **BBPF**: book preferred company, otherwise book cheapest flight
Experimental Results with 1 param. using DFS (in Secs.) ...

<table>
<thead>
<tr>
<th>#Co.</th>
<th>BPF</th>
<th>BMₓF</th>
<th>BPMₓF</th>
<th>BBF</th>
<th>BPBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.02</td>
<td>0.10</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>0.35</td>
<td>1.58</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>53.99</td>
<td>259.39</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>0.02</td>
<td>0.02</td>
<td>&gt; tₓmax</td>
<td>&gt; tₓmax</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>0.04</td>
<td>0.04</td>
<td>&gt; tₓmax</td>
<td>&gt; tₓmax</td>
</tr>
<tr>
<td>20</td>
<td>0.04</td>
<td>0.05</td>
<td>0.06</td>
<td>&gt; tₓmax</td>
<td>&gt; tₓmax</td>
</tr>
<tr>
<td>50</td>
<td>0.31</td>
<td>0.51</td>
<td>0.60</td>
<td>&gt; tₓmax</td>
<td>&gt; tₓmax</td>
</tr>
<tr>
<td>100</td>
<td>2.47</td>
<td>3.15</td>
<td>3.43</td>
<td>&gt; tₓmax</td>
<td>&gt; tₓmax</td>
</tr>
</tbody>
</table>

Results with 1 explicit parameter: company
PKS v0.6-alpha-2 (Linux)
(tₓmax = 300secs.)
... Experimental Results with 5 param. using DFS (in Secs.)

<table>
<thead>
<tr>
<th>#Co.</th>
<th>BPF</th>
<th>BMxF</th>
<th>BPMxF</th>
<th>BBF</th>
<th>BPBF</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.17</td>
<td>0.21</td>
<td>0.37</td>
<td>1.27</td>
<td>8.42</td>
</tr>
<tr>
<td>3</td>
<td>0.58</td>
<td>0.72</td>
<td>1.20</td>
<td>24.02</td>
<td>109.33</td>
</tr>
<tr>
<td>4</td>
<td>1.45</td>
<td>2.20</td>
<td>3.71</td>
<td>&gt; t&lt;sub&gt;max&lt;/sub&gt;</td>
<td>&gt; t&lt;sub&gt;max&lt;/sub&gt;</td>
</tr>
<tr>
<td>5</td>
<td>3.76</td>
<td>4.33</td>
<td>4.65</td>
<td>&gt; t&lt;sub&gt;max&lt;/sub&gt;</td>
<td>&gt; t&lt;sub&gt;max&lt;/sub&gt;</td>
</tr>
<tr>
<td>10</td>
<td>80.60</td>
<td>96.45</td>
<td>105.49</td>
<td>&gt; t&lt;sub&gt;max&lt;/sub&gt;</td>
<td>&gt; t&lt;sub&gt;max&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

Results with 5 explicit parameters:
- company, origin, destination, departure date, and arrival date
- PKS v0.6-alpha-2 (Linux)
- \( t_{\text{max}} = 300 \text{secs.} \)
Advantages of Our Approach

- Modularity and re-usability

- Can handle cases that previous approaches cannot (e.g., physical actions having direct effect on sensing actions) \([\text{McIlraith \& Son, 2002}]\)

- No need for pre-specified generic plans
Open Problems and Future Work

- Customizing domain theory based on problem
  - DSURs generation (desc. goal + user pref.)
- Large search space, off-line
- Optimization constraints problems do not scale up well
- Representation of atomic services
- Translation of OWL, DAML-S, etc. into PKS/Golog specifications

**IG-JADE-PKSlab** toolkit
- Experiments + case studies to validate performance and scalability of integrated framework
- Plan execution and contingency recovery
Thank You!