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 *[Fikes & Nilsson, 1971]*
  - Four Dbs: $\boldsymbol{K_F}$, $\boldsymbol{K_W}$, $\boldsymbol{K_V}$, $\boldsymbol{K_X}$
  - Actions:

Action	Precondition	Effects
checkFlightSpace(n,d)	*K(existsFlight(n,d))*	*add(Kw, availFlight(n,d))*

  - DSURs: *K(existsFlight(n,d))* $\Rightarrow$ *add(Kv, flightNum(n,d))*
  - Goal
  - Planning problem: **< *I*, *A*, *U*, *G* >**

WSC in PKS

- PKS primitive actions correspond to WS
  - Knowledge-producing actions $\leftrightarrow$ information gathering WS
  - Physical actions $\leftrightarrow$ world-altering WS
- New WS becomes available $\rightarrow$ add new primitive action to domain specification

WS Representation in PKS

- For each action $\boldsymbol{A_i}$:
  - encode user preferences / customization constraints using $\boldsymbol{desA_i}$ fluent *[McIlraith & Son, 2002]*
  - encode domain specific search control constraints using $\boldsymbol{indA_i}$ 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))* *K(indFindRFlight(x))* *K(desFindRFlight(x))*	*add(Kw, flightExists(x))* *add(Kf,¬indFindRFlight(x))*
bookFlight(x)	*K(airCo(x))* *K(availFlight(x))* *K(indBookFlight(x))* *K(desBookFlight(x))*	*add(Kf, bookedFlight(x))* *del(Kf, availFlight(x))* *add(Kf,¬indBookFlight(x))*

Domain specific update rules (DSUR)

$$K(airCo(x)) \wedge \neg Kv(flightNum(x)) \wedge K(flightExists(x)) \Rightarrow add(Kv, flightNum(x))$$

1 explicit parameter: *company*

PKS Goal Example (Prob. BMxF)

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

/* book company within budget */

$\exists_K(x)\ [K(airCo(x)) \wedge K(bookedFlight(x)) \wedge$

$K(\neg priceGtMax(x))]$ |

/* no flight booked */

KnowNoBudgetFlight |

KnowNoAvailFlight |

KnowNoFlightExists

User Pref. / Customization Constraints Example (Prob. BMxF)

- DSUR 1:

$$K(airCo(x)) \wedge \neg Kw(priceGtMax(x)) \wedge Kv(userMaxPrice) \wedge Kv(flightCost(x)) \Rightarrow add(Kw, priceGtMax(x))$$

- DSUR 2:

$$K(airCo(x)) \wedge K(\neg priceGtMax(x)) \wedge \neg Kw(desBookFlight(x)) \Rightarrow add(Kf, desBookFlight(x))$$

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.) ...

#Co.	BPF	BM_xF	BPM_xF	BBF	BPBF
2	0.00	0.00	0.00	0.02	0.10
3	0.00	0.01	0.01	0.35	1.58
4	0.00	0.01	0.01	53.99	259.39
5	0.01	0.02	0.02	$> t_{max}$	$> t_{max}$
10	0.01	0.04	0.04	$> t_{max}$	$> t_{max}$
20	0.04	0.05	0.06	$> t_{max}$	$> t_{max}$
50	0.31	0.51	0.60	$> t_{max}$	$> t_{max}$
100	2.47	3.15	3.43	$> t_{max}$	$> t_{max}$

Results with 1 explicit parameter: *company*
PKS v0.6-alpha-2 (Linux)
(t_{max} = 300secs.)

... Experimental Results with 5 param. using DFS (in Secs.)

#Co.	BPF	BM_xF	BPM_xF	BBF	BPBF
2	0.17	0.21	0.37	1.27	8.42
3	0.58	0.72	1.20	24.02	109.33
4	1.45	2.20	3.71	$> t_{max}$	$> t_{max}$
5	3.76	4.33	4.65	$> t_{max}$	$> t_{max}$
10	80.60	96.45	105.49	$> t_{max}$	$> t_{max}$

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

Advantages of Our Approach

- Modularity and re-usability
- Can handle cases that previous approaches can-not (e.g., physical actions having direct effect on sensing actions) *[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-PKSlib*** toolkit
- Experiments + case studies to validate performance and scalability of integrated framework
- Plan execution and contingency recovery

Thank You!