LotTraveler: flexible on-line job management with ASP

Harald Beck    Minh Dao-Tran    Thomas Eiter
Konstantin Schekotihin

Stream Reasoning Workshop 2016
Agenda

1. Motivation

2. Requirements to KRRS

3. Use Cases

4. Outlook
Production 4.0 @ Infineon Austria

- Fault Analysis Lab (FA) – fault localization in integrated circuits:
  - development and production
  - customer returns

© Infineon
LotTraveler

1. Motivation

Production 4.0 @ Infineon Austria

- Fault Analysis Lab (FA) – fault localization in integrated circuits:
  - development and production
  - customer returns

Agile management – increase throughput by identifying and eliminating waste
Production 4.0 @ Infineon Austria

- Fault Analysis Lab (FA) – fault localization in integrated circuits:
  - development and production
  - customer returns

Agile management – increase throughput by identifying and eliminating waste

- The types of waste include, for example:
  - overconsumption of resources, like employees, tools, or materials
  - unreasonable workflows, e.g. redundant investigation steps, insufficient sequences, etc.
  - irregularity of tasks – replace FIFO by JIT
LotTraveler Overview

An intelligent system providing support for:
- planning and monitoring of analysis tasks
- disruption management
- employee coordination by group leaders
- overall lab management and reporting
Requirements to KRRS

- Ability to process streams of facts
- Express statements about time
- Represent and solve planning/scheduling problems
- Maintenance of KB consistency
- Human-readable syntax and clear semantics
- Fast update of KB with new rules
Normal Operation

- One shift in the lab is over and a group leader generates a work plan for a new shift

Example

- **Given**: sets of employees, tools and job tasks
- **Find**: an assignment of tasks to employees and tools such that a set of requirements is satisfied

```
assign(E, T, M, N) ← not not assign(E, T, M, N), taskM(T, M), empl(E), @N free(E)
@N+L finish(E, T, M) ← assign(E, T, M, N), duration(T, L), not □L ◇ done(T)
← assign(E1, T, _, _), assign(E2, T, _, _), E1 ≠ E2
← assign(E, T, M, _), costE(E, C), costT(T, K) [K@C, E, T]
```
3. Use Cases

Disruption Management I

- In many cases the expected and actual states might be different.
- The KRRS must take the new information into account and maintain the plan consistency.

Example

- Processing of a task might take longer than expected.

\[
@_N\text{replan}(E, T, M) \leftarrow @_N\text{finish}(E, T, M), \text{now}(N'), \\
N' > N, \text{not } \Diamond \text{done}(T)
\]
Disruption Management II

Example

- A tool may be broken or malfunctioning

\[ \Box^{N+Z} \Diamond cancel(M) \leftarrow @_N broken(M), \, expectedTime(M, Z) \]

- Non-productive interruptions like meetings may take place

\[ \Box^{N+D} \Diamond \neg free(E) \leftarrow @_N meeting(E, D) \]

- Jobs can require special handling

\[ cancel(T_1) \leftarrow rocket(T), \, assign(E, T, M, N), \]
\[ \Box^+ N \, \Diamond finish(\_, T_1, M), \, T_1 \neq T \]
Outlook

Current work:
- Ongoing implementation of LotTraveler @ Infineon Austria
- Definition and evaluation of planning models for normal operation
- Finalization of statistical analysis of FA Lab events

Open questions for KRRS:
- Automated vs. semi-automated repair of a plan
- Declarative specification of preferences for a repair
- Performance of repair algorithms