Planning@SAP: An Application in Business Process Management

Jörg Hoffmann[1] Ingo Weber[2] Frank Michael Kraft[3]

[1]SAP Research, Karlsruhe, Germany

[2]UNSW, Sydney, Australia (work performed while at SAP Research)

[3]SAP, Walldorf, Germany

"Deployed application"? Hm, well . . .

Outline

- Business Process Management & Planning
- PDDL@SAP: The SAM Model of Transactions
- Planning Formalization
- Algorithms & Results
- Discussion

Outline

Business Process Management

- **What is a Business Process?**

  "A business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal."

  - User-friendly presentation in terms of diagrams
  - Control-flow, formalized e.g. as Petri-net

- **What is Business Process Management (BPM)?**
  - Understand, configure, monitor, implement, ...
  - Central activity in BPM: modeling (create/adapt processes)

Planning@BPM

- **Mind the Gap!**
  - Business-view of process vs. IT-implementation of process
  - Takes time and resources (call your local SAP consultant)
- **Planning@BPM**
  - Describe each IT transaction by precond/postcond
  - Business user enters “goal” (what do I want done here?)
- **Planning Requirements**
  - Instant response times
  - Low modeling overhead

Planning@BPM

- **Mind the Gap!**
  - Business-view of process vs. IT-implementation of process
  - Takes time and resources (call your local SAP consultant)
- **Planning@BPM**
  - Describe each IT transaction by precond/postcond
  - Business user enters “goal” (what do I want done here?)
- **Planning Requirements**
  - Instant response times
  - Low modeling overhead
    ($\Longrightarrow$ business user enters goal?)

Planning@BPM

- **Mind the Gap!**
  - Business-view of process vs. IT-implementation of process
  - Takes time and resources (call your local SAP consultant)
- **Planning@BPM**
  - Describe each IT transaction by precond/postcond
  - Business user enters “goal” (what do I want done here?)
- **Planning Requirements**
  - Instant response times
  - Low modeling overhead
    ($\Longrightarrow$ business user enters goal? pre/post??)

Outline

- Business Process Management & Planning
- **PDDL@SAP: The SAM Model of Transactions**
- Planning Formalization
- Algorithms & Results
- Discussion

Verbatim from SAP

```
<StatusAndAction><StatusSchema Name="CustomerInvoice__Standard-SAM" ...
<StatusVariableOccurrence Name="Consistency" ...
<Enable StatusValueOccurrence= ... ActionOccurrence= ...
<Required StatusValueOccurrence= ... ActionOccurrence= ...
</StatusAndAction>
```

Verbatim from SAP

```
<StatusAndAction><StatusSchema Name="CustomerInvoice__Standard-SAM" ...
<StatusVariableOccurrence Name="Consistency" ...
<Enable StatusValueOccurrence= ... ActionOccurrence= ...
<Required StatusValueOccurrence= ... ActionOccurrence= ...
</StatusAndAction>
```

Status and Action Management (SAM)

- **Scope of the Model**
  - Behavior of business objects (e.g. “Customer Quote”) . . .
  - . . . over IT transactions (e.g. “Submit for Approval”)
  - Domain & language of business users!
- **Structure of the Model**
  - Business object properties: status variables
  - IT transactions: precondition/postcondition (propositional logic)
- **Original Use of the Model**
  - Declarative preconditions check (avoid implementation bugs)
  - To some extent: generate application code from changes to action set

SAM Example, and 2PDDL

Action name	precondition	postcondition
Check CQ Completeness	CQ.archivation:notArchived(x)	CQ.completeness:complete(x) OR CQ.completeness:notComplete(x)
Check CQ Consistency	CQ.archivation:notArchived(x)	CQ.consistency:consistent(x) OR CQ.consistency:notConsistent(x)
Check CQ Approval Status	CQ.archivation:notArchived(x) AND CQ.completeness:complete(x) AND CQ.consistency:consistent(x)	CQ.approval:Necessary(x) OR CQ.approval:notNecessary(x)
CQ Approval	CQ.archivation:notArchived(x) AND CQ.approval:Necessary(x)	CQ.approval:granted(x)
Submit CQ	CQ.archivation:notArchived(x) AND (CQ.approval:notNecessary(x) OR CQ.approval:granted(x))	CQ.submission:submitted(x)
Mark CQ as Accepted	CQ.archivation:notArchived(x) AND CQ.submission:submitted	CQ.acceptance:accepted(x)
Create Sales Order from CQ	CQ.archivation:notArchived(x) AND CQ.acceptance:accepted(x)	CQ.followUp:documentCreated(x) (*)
Archive CQ	CQ.archivation:notArchived(x)	CQ.archivation:Archived(x)

2PDDL:

- Initial state: defined in SAM for every business object
- Goal: status variable value(s) given by business user
- Actions: preconditions as usual
  disjunctive effects: non-deterministic, immediately observed

Outline

Planning Formalization

Planning task (X, dA, ndA, I, G): X finite domain; dA, ndA: pre_a, eff_a sets of facts; I complete, G set of facts. State s assignment, $s \oplus f$ applies f to s, T tree of actions.

T solves (s, ndA_{av}) iff either:

1. T is empty and $G \subseteq s$; or
2. root of T is $a \in dA$, a applicable in s, a has exactly one son, tree rooted at that son solves $(s \oplus eff_a, ndA_{av})$; or
3. root of T is $a \in ndA_{av}$, a applicable in s, a has one son for every eff_a, each $(s \oplus eff_a, ndA_{av} \setminus \{a\})$ is either (i) unsolvable or (ii) solved by respective sub-tree of T, where (ii) is the case for at least one son.

Plan: tree that solves (I, ndA).

Planning Formalization

Non-deterministic effects vs. observations:

T solves (s, ndA_{av}) iff either:

1. T is empty and $G \subseteq s$; or
2. root of T is $a \in dA$, a applicable in s, a has exactly one son, tree rooted at that son solves $(s \oplus eff_a, ndA_{av})$; or
3. root of T is $a \in ndA_{av}$, a applicable in s, a has one son for every eff_a, each $(s \oplus eff_a, ndA_{av} \setminus \{a\})$ is either (i) unsolvable or (ii) solved by respective sub-tree of T, where (ii) is the case for at least one son.

Plan: tree that solves (I, ndA).

- No distinction between non-deterministic effects and observations
- $\Longrightarrow$ no non-trivial beliefs, state knowledge remains exact

Planning Formalization

Upper bound on repeating non-deterministic actions:

T solves (s,ndA_{av}) iff either:

1. T is empty and $G \subseteq s$; or
2. root of T is $a \in dA$, a applicable in s, a has exactly one son, tree rooted at that son solves $(s \oplus eff_a, ndA_{av})$; or
3. root of T is $a \in ndA_{av}$, a applicable in s, a has one son for every eff_a, each $(s \oplus eff_a, ndA_{av} \setminus \{a\})$ is either (i) unsolvable (*) or (ii) solved by respective sub-tree of T, where (ii) is the case for at least one son.

Plan: tree that solves (I, ndA).

- Blind repetition not useful (check and re-check completeness . . .)
- With infinite repetition: recursion (*) may result in same task $\Longrightarrow$ ex. task that is solvable iff it is unsolvable

Planning Formalization

Failed nodes:

T solves (s, ndA_{av}) iff either:

1. T is empty and $G \subseteq s$; or
2. root of T is $a \in dA$, a applicable in s, a has exactly one son, tree rooted at that son solves $(s \oplus \mathit{eff}_a, ndA_{av})$; or
3. root of T is $a \in ndA_{av}$, a applicable in s, a has one son for every eff_a, each $(s \oplus \mathit{eff}_a, ndA_{av} \setminus \{a\})$ is either (i) unsolvable or (ii) solved by respective sub-tree of T, where (ii) is the case for at least one son.

Plan: tree that solves (I, ndA).

- Allow-failure vs. more-complex-goals: c.f. business user ...
- $\Longrightarrow$ no explicit failure handling in the constructed processes

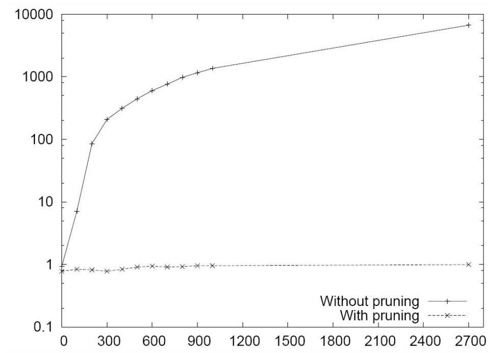
Outline

- Business Process Management & Planning
- PDDL@SAP: The SAM Model of Transactions
- Planning Formalization
- **Algorithms & Results**
- Discussion

Planning Algorithms

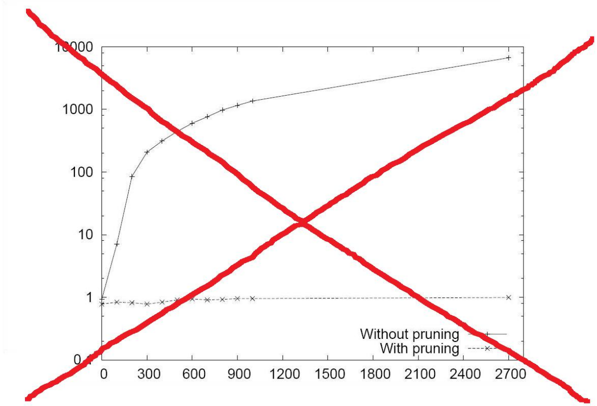
- **Forward AND-OR Search Variant:**
  - OR nodes: states, OR'ed children are applicable actions
  - AND nodes: actions, AND'ed children are different outcomes
  - AND node solved iff ((all children solved or failed), ≥ 1 solved)
- **Relaxed Plans Heuristic:**
  - Act as if each non-deterministic outcome is separate action
  - May return ∞ as usual $\implies$ main source of "failed" ...!

Runtime Performance



- $x = \#$(input-services not contained in solution): of 2700 possible ones
- SAM currently omits many dependencies across business objects

Runtime Performance



- $x = \#$(input-services not contained in solution): of 2700 possible ones
- SAM currently omits many dependencies across business objects

Runtime Performance

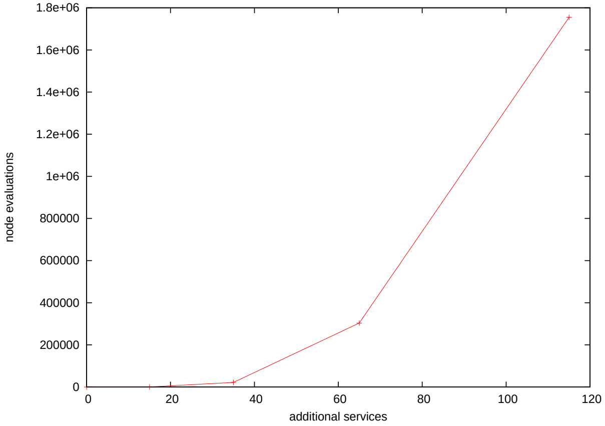
```
./ff -p ../../input/domains/spark09example/ -o domain-Full.pddl -f facts-Full.pddl

                                   ...

time spent:    1.31 seconds instantiating 1384944 easy, 0 hard action templates
               0.41 seconds reachability analysis, yielding 110956 facts and 2616 actions
               0.01 seconds creating final representation with 4354 relevant facts
               0.02 seconds building connectivity graph
               0.01 seconds searching, evaluating 7 states, to a max depth of 4
                  0 actions skipped due to simple duplicate detection
               1.76 seconds total time
```

- Transformation SAM2PDDL; anonymized PDDL will (likely) be publicly available
- FF extension ("SAP-FF") vastly more efficient than previous tool

return $(h(s) == \infty \; ? \; \infty : 1)$



- $x = \#$(input-services not contained in solution): of 2700 possible ones
- Without any h: out of memory at $x = 20$: finding "failed nodes" is difficult . . .

Outline

Discussion

- Technique implemented within SAP NetWeaver platform
- Currently in **INITIAL** steps towards **PILOT** customer evaluation

- Algorithm performance good based on easy modifications of FF
- SAM currently omits many dependencies across business objects
- Richer model currently under development $\Longrightarrow$ further extend FF

- Handle plan preferences

... pre/post-based description of IT transactions is NOT only an academic exercise!