Attribute Grammars for Modeling Planning Domains

Roman Barták
Charles University, Czech Republic
with contributions from **Adrien Maillard**, **Rafael C. Cardoso**

Model vs. model-free approaches

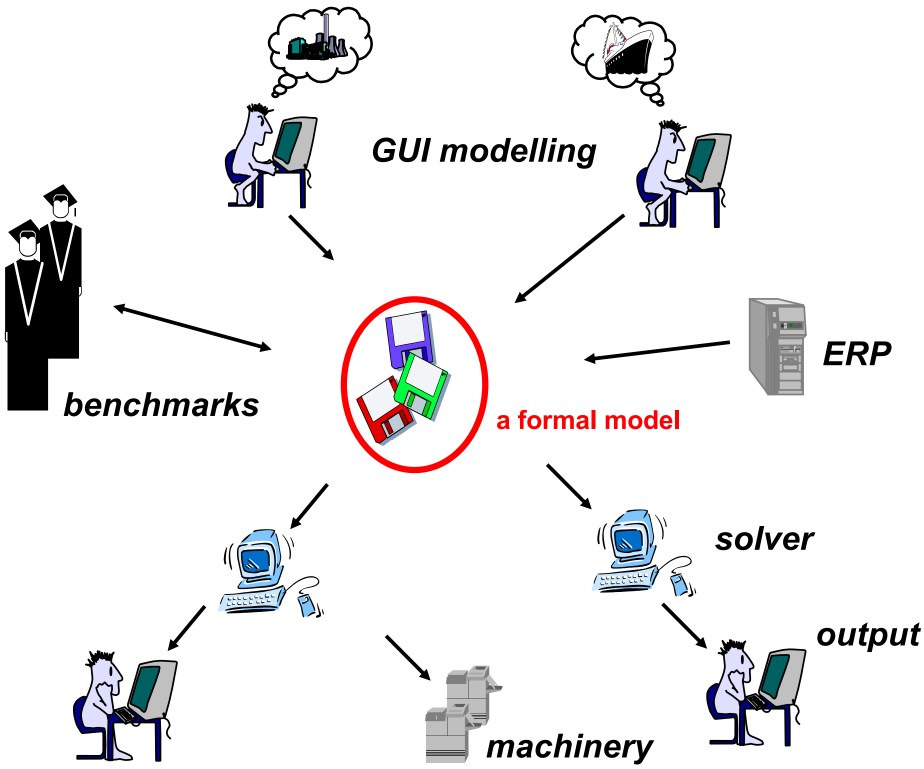
Model-free approaches

- easy to use (+)
- good results (+)
- black box (-)
- „strange“ mistakes (-)

Model-based approaches

- more complex design of models (-)
- explanation of results (+)
- model verification (+)

A model centric approach [KEPS 2003]



Attribute grammars

a **context-free** (CF) grammar, where the symbols are annotated by sets of **attributes** that can be connected via **constraints**

$a^i b^j c^k$ (CF grammar)

S → A.B.C
A → a | a.A
B → b | b.B
C → c | c.C

$a^i b^i c^i$ (attribute grammar)

S(n) → A(n_a).B(n_b).C(n_c)	[$n=n_a=n_b=n_c$]
A(n) → a	[n=1]
A(n) → a.A(m)	[n=m+1]
B(n) → b	[n=1]
B(n) → b.B(m)	[n=m+1]
C(n) → c	[n=1]
C(n) → c.C(m)	[n=m+1]

Outline

Part I. Translating HTNs to Attribute Grammars

- *Hierarchical Task Networks*
- *Attribute Grammars with Set Attributes (Timelines)*
- *Translation Procedure*

Part II. Validation of Hierarchical Plans using Grammars

- *Parsing of Attribute Grammars*

Outline

Part I. Translating HTNs to Attribute Grammars

- *Hierarchical Task Networks*
- *Attribute Grammars with Set Attributes (Timelines)*
- *Translation Procedure*

Part II. Validation of Hierarchical Plans using Grammars

- *Parsing of Attribute Grammars*

Motivation

Can we represent existing planning domain modeling formalisms as an attribute grammar?

Why?

- provide a unifying framework for modeling planning domains and problems that can be used for **domain model verification**, **plan and goal recognition**, **plan validation**, **domain model acquisition**, as well as for **efficient planning**

Why attribute grammars?

- to exploit existing techniques from formal languages

Automated planning

find a sequence of actions (a plan) to achieve some goal (solve a task)

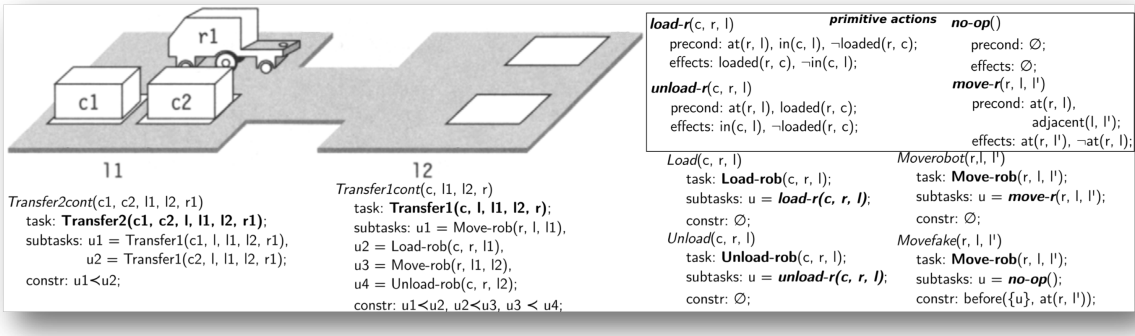
uses action model with preconditions and effects

```
Load-r(container, robot, location)
    Pre:  in(container, location), at(robot, location),
          not loaded(robot, container)
    Eff:  not in(container, location), loaded(robot, container)
```

HTN

Hierarchical Task Networks seem like a natural candidate to translate to attribute grammars.

- a hierarchical structure
- some extra ordering constraints

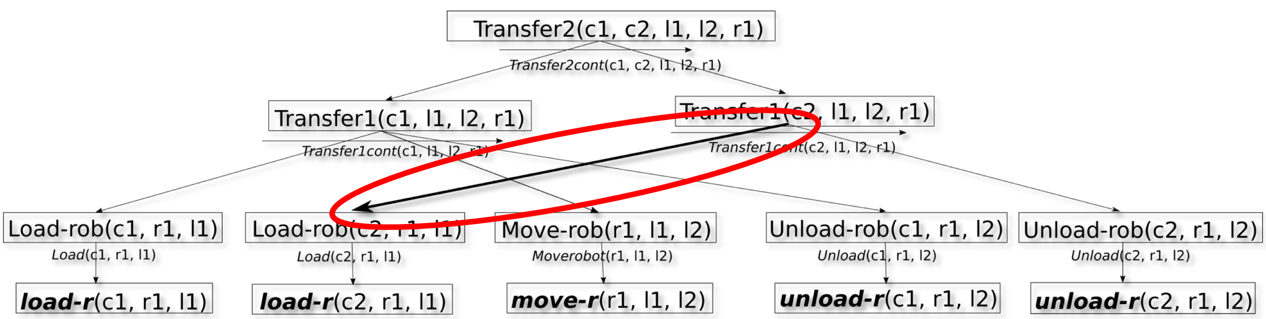


Grammars have been used to represent HTN

Can they cover HTN fully?

HTN – activity interleaving

No existing grammar formalism used to encode HTNs supports task interleaving!



Set attributes

Assume that we want to assign indexes to symbols in $a^i b^i c^i$ such that any permutation of indexes is allowed such as $a_1b_2c_3$, $a_1b_3c_2$, $a_2b_1c_3$ *etc.*

$S(n,I) \rightarrow A(n_a,I_a).B(n_b,I_b).C(n_c,I_c)$	$[n=n_a=n_b=n_c,\ I = I_a \cup I_b \cup I_c,$ $dom(I,1,3n),\ allDiff(I)]$
$A(n,I) \rightarrow a(i)$	$[n=1,\ I=\{i\}]$
$A(n,I) \rightarrow a(i).A(m,I')$	$[n=m+1,\ I=\{i\} \cup I']$
$B(n,I) \rightarrow b(i)$	$[n=1, I=\{i\}]$
$B(n,I) \rightarrow b(i).B(m,I')$	$[n=m+1,\ I=\{i\} \cup I']$
$C(n,I) \rightarrow c(i)$	$[n=1,\ I=\{i\}]$
$C(n,I) \rightarrow c(i).C(m,I')$	$[n=m+1,\ I=\{i\} \cup I']$

Timeline constraint

Actions generated during decomposition introduce "before" (precondition) and "after" (effect) events.

The **timeline constraint** ensures the correct ordering of events.

position action	0	1 load-r(c1, r1, l1)	2 load-r(c2, r1, l1)	3 move-r(r1, l1, l2)	4 unload-r(c1, r1, l2)	5 unload-r(c2, r1, l2)
loaded(r1, c1)	a^-	$b^- a^+$			$b^+ a^-$	
loaded(r1, c2)	a^-		$b^- a^+$			$b^+ a^-$
at(r1, l1)	a^+	b^+	b^+	$b^+ a^-$		
at(r1, l2)	a^-			a^+	b^+	b^+
in(c1, l1)	a^+	$b^+ a^-$				
in(c1, l2)	a^-				a^+	
in(c2, l1)	a^+		$b^+ a^-$			
in(c2, l2)	a^-					a^+

HTN as an attribute grammar
principles

Rewriting rules describe task decompositions, the timeline constraint orders the actions.

Initialisation

$\mathbf{S(S_0) \rightarrow TN_0(I,TL')}$ [n = |I|, dom(I, 1, n), allDiff(I), TL = TL′ ∪ InitEvents(S_0), Timeline(TL)]

Task network

$\mathbf{TN_0(I,TL) \rightarrow T_1(I_1,TL_1)...T_m(I_m,TL_m)}$ [C]

Task decomposition to task networks

$\mathbf{T_k(I, TL) \rightarrow TN_{k1}(I, TL) \mid \ldots \mid TN_{kn}(I, TL)}$ []

Primitive task (action)

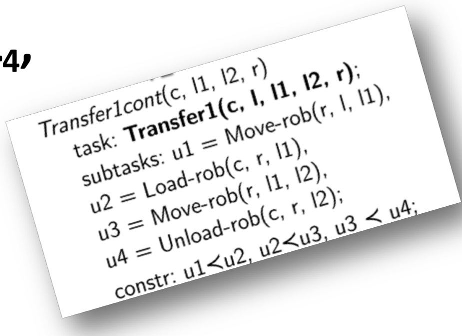
$\mathbf{T_k(I,TL) \rightarrow a_k(i)}$ [I = {i}, TL = events(a_k,i)]

HTN as an attribute grammar
example

$\mathbf{Transfer1_{c,l,l1,l2,r}(I,TL) \rightarrow Transfer1cont_{c,l,l1,l2,r}(I,TL)}$

$\mathbf{Transfer1cont_{c,l,l1,l2,r}(I,TL) \rightarrow Move\text{-}rob_{r,l,l1}(I_1,TL_1).\ Load\text{-}rob_{c,r,l1}(I_2,TL_2).\ Move\text{-}rob_{r,l1,l2}(I_3,TL_3).\ Unload\text{-}rob_{c,r,l2}(I_4,TL_4)}$ **[C]**

where **C = {**$\mathbf{TL = TL_1 \cup TL_2 \cup TL_3 \cup TL_4,}$
$\mathbf{I = I_1 \cup I_2 \cup I_3 \cup I_4,}$
$\mathbf{max(I_1) < min(I_2),}$
$\mathbf{max(I_2) < min(I_3),}$
$\mathbf{max(I_3) < min(I_4)\}}$



Outline

Part I. Translating HTNs to Attribute Grammars

- *Hierarchical Task Networks*
- *Attribute Grammars with Set Attributes (Timelines)*
- *Translation Procedure*

Part II. Validation of Hierarchical Plans using Grammars

- *Parsing of Attribute Grammars*

Motivation

Validate compliance of a given plan with respect to an HTN model.

Why is it **important**?

- competitions of HTN planners
- soundness of planners and HTN models
- first step towards plan/goal recognition

What is **novel** there?

- the first approach that covers full HTN

Task decomposition as a grammar rule

Transfer1(c, l1, l2, r) → Load-rob(c, r, l1).
Move-rob(r, l1, l2).
Unload-rob(c, r, l2) [C]

C = {Load-rob ≺ Move-rob,
Move-rob ≺ Unload-rob,
before({Load-rob}, at(r, l1)),
before({Load-rob}, at(c, l1)),
between({Load-rob},{Move-rob}, at(r, l1)),
between({Move-rob},{Unload-rob}, at(r, l2)),
between({Load-rob},{Unload-rob}, in(c, r))}

HTN plan validation via parsing

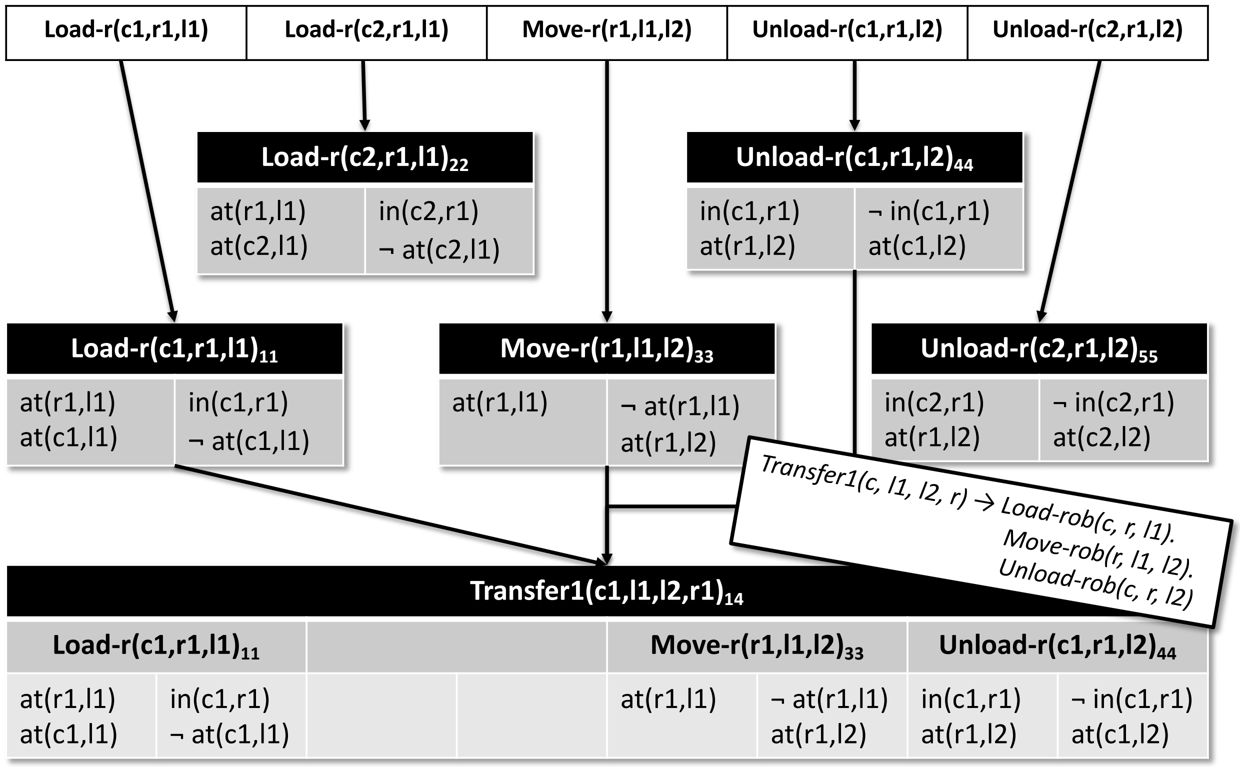
Parsing is a bottom-up approach that groups actions (terminals) to tasks (non-terminals).

For each derived task, we keep a **timeline** – a sequence of slots containing:

- actions (some slots may be empty)
- known effects and partially specified states

Rule is fired when **matching tasks** for the right-hand side are found and their **timelines** can be properly **merged**.

Example of parsing step



Example of parsing step

Timeline merging

Transfer1(c1,l1,l2,r1)$_{14}$							
Load-r(c1,r1,l1)$_{11}$				Move-r(r1,l1,l2)$_{33}$		Unload-r(c1,r1,l2)$_{44}$	
at(r1,l1) at(c1,l1)	in(c1,r1) ¬ at(c1,l1)			at(r1,l1)	¬ at(r1,l1) at(r1,l2)	in(c1,r1) at(r1,l2)	¬ in(c1,r1) at(c1,l2)

before({Load-rob}, at(r1, l1)),
before({Load-rob}, at(c1, l1)),
between({Load-rob},{Move-rob}, at(r1, l1)),
between({Move-rob},{Unload-rob}, at(r1, l2)),
between({Load-rob},{Unload-rob}, in(c1, r))}

Add method constraints

		at(r1,l1) in(c1,r1)		in(c1,r1)			

$$\mathrm{Pre}^+_{i+1} = (\mathrm{Pre}^+_i \setminus \mathrm{Post}^-_i) \cup \mathrm{Post}^+_i$$
$$\mathrm{Pre}^-_{i+1} = (\mathrm{Pre}^-_i \setminus \mathrm{Post}^+_i) \cup \mathrm{Post}^-_i$$

Propagate states

		¬ at(c1,l1)				¬ at(r1,l1)	

The algorithm

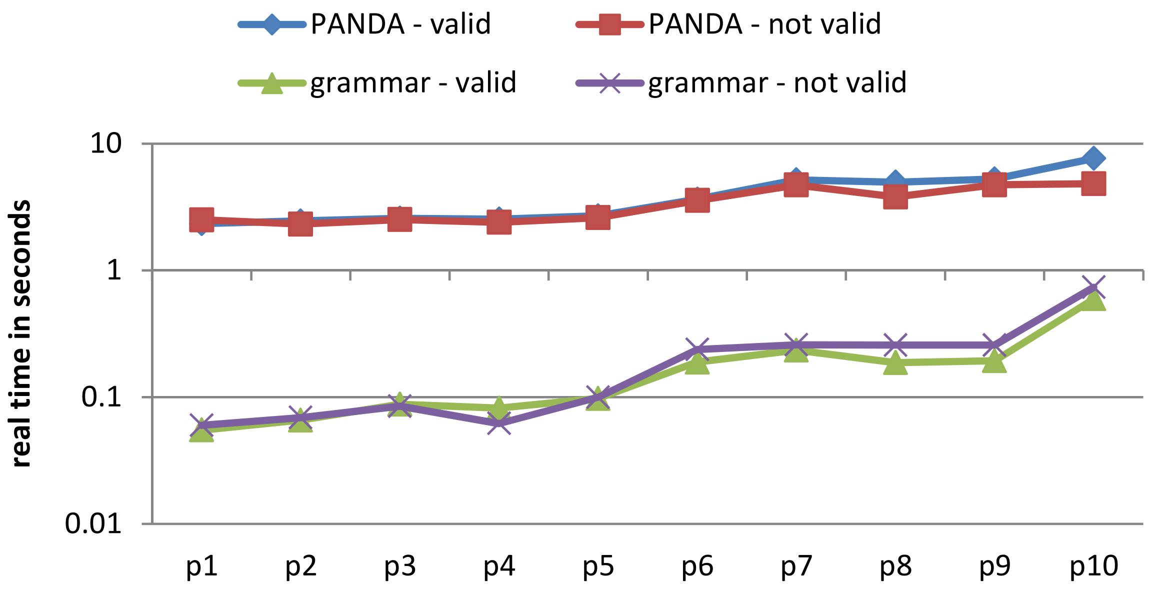
Experiments

Comparison with PANDA, currently the only HTN plan validator (based on SAT)

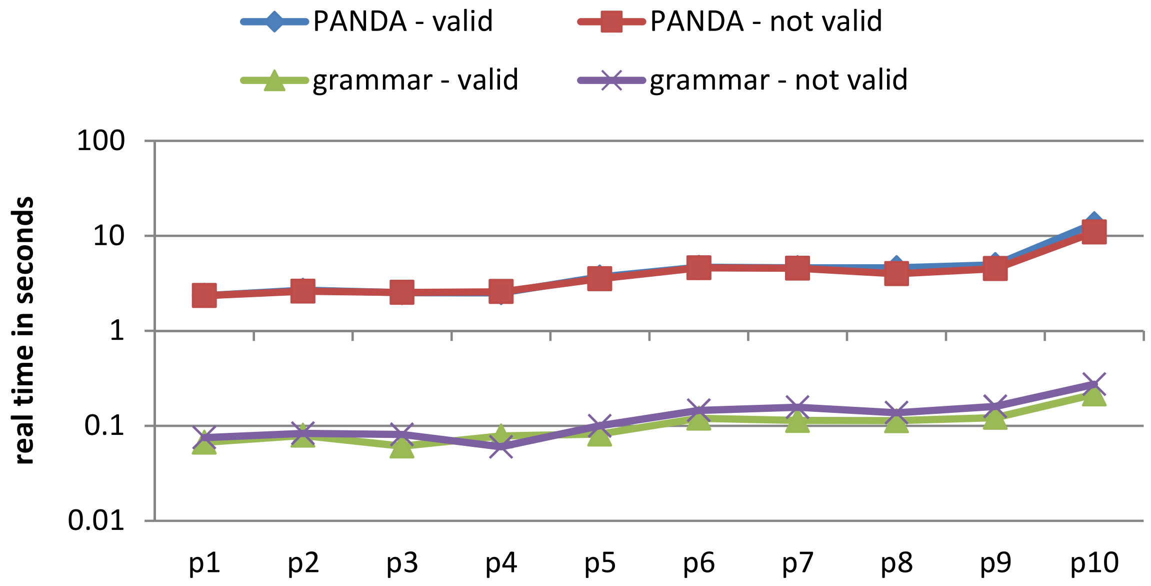
- PANDA compiles away some before constraints but cannot handle the other method causal constraints

- Two planning domains
  - Satellite and Transport
- Measuring runtime
  - for correct plans and for wrong plans

Results of experiments - Satellite



Results of experiments - Transport



Summary

STRIPS, HTN with task insertions, and procedural domain control knowledge can be fully **automatically translated** to attribute grammars

– Roman Barták, Adrien Maillard:
  Attribute grammars with set attributes and global constraints as a unifying framework for planning domain models. PPDP 2017: 39-48

HTN plans can be **fully validated** with respect to the HTN model using attribute grammars

– Roman Barták, Adrien Maillard, Rafael Cauê Cardoso:
  Validation of Hierarchical Plans via Parsing of Attribute Grammars. ICAPS 2018: 11-19

Possible next steps

- identifying a specific bug in plan
- working with partial input plans (plan/goal recognition)
  - plan prefix
  - missing, extra (non-related), wrong observations
  - predicting the next action
- modifying the model to comply with observations
- verifying internal consistency of HTN models

Roman Barták
Charles University, Faculty of Mathematics and Physics
bartak@ktiml.mff.cuni.cz