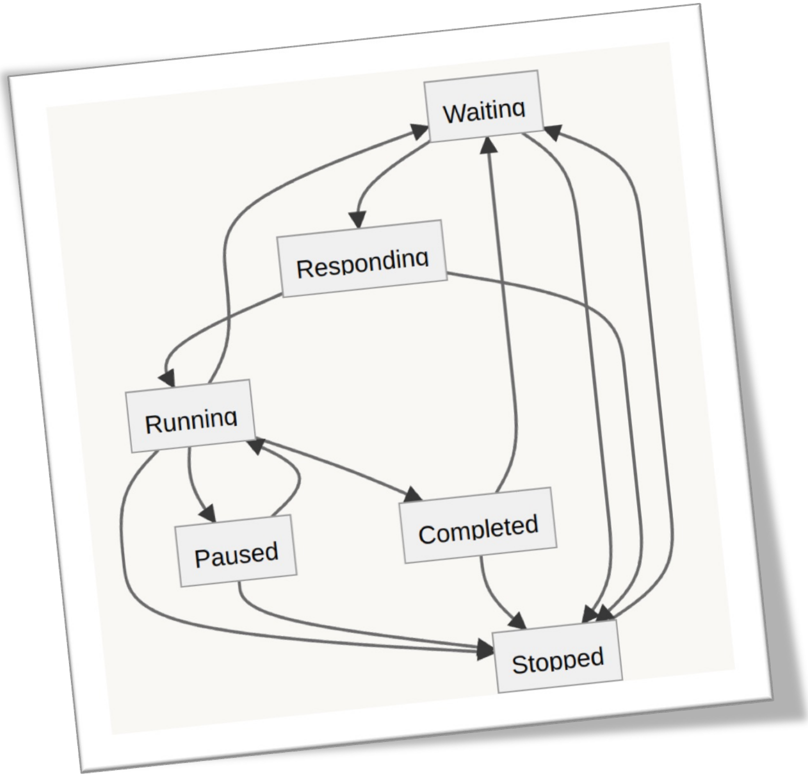
Hierarchical Planning
On Solving Planning Problems by Task Decompositions and Beyond

Roman Barták

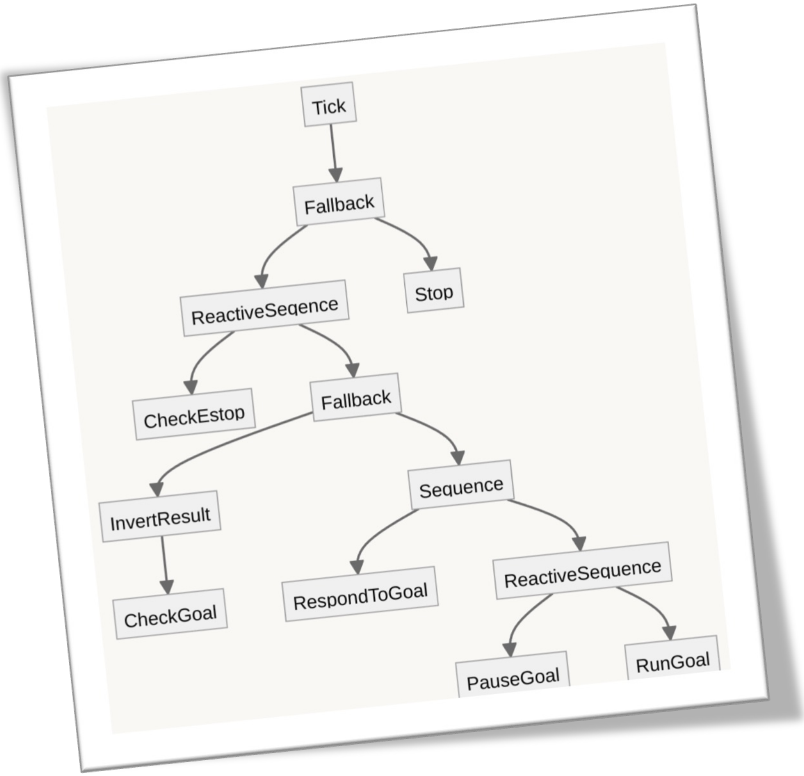
Charles University, Faculty of Mathematics and Physics
Czech Republic

Models of Agent Behaviour

Finite state machines

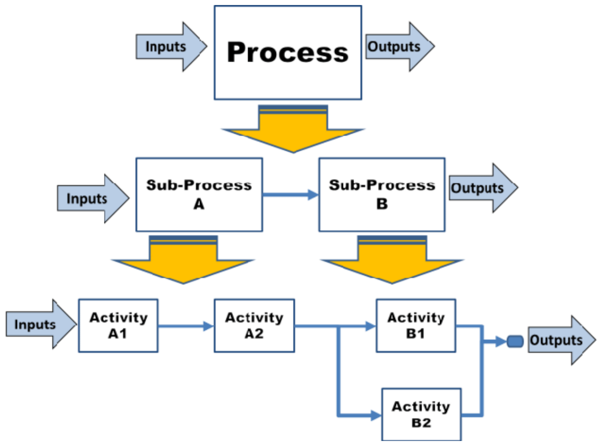


Behaviour Trees

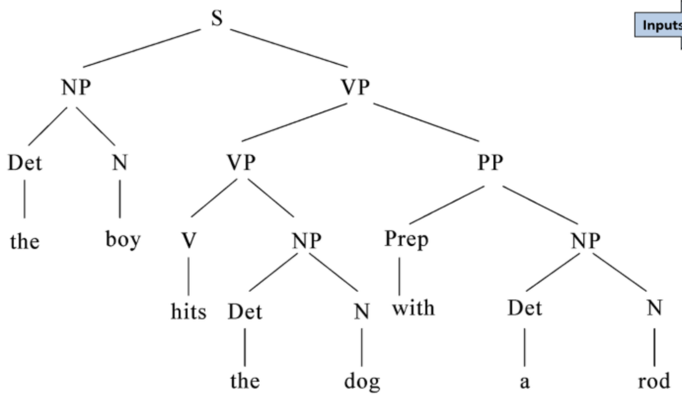


Hierarchical Representations

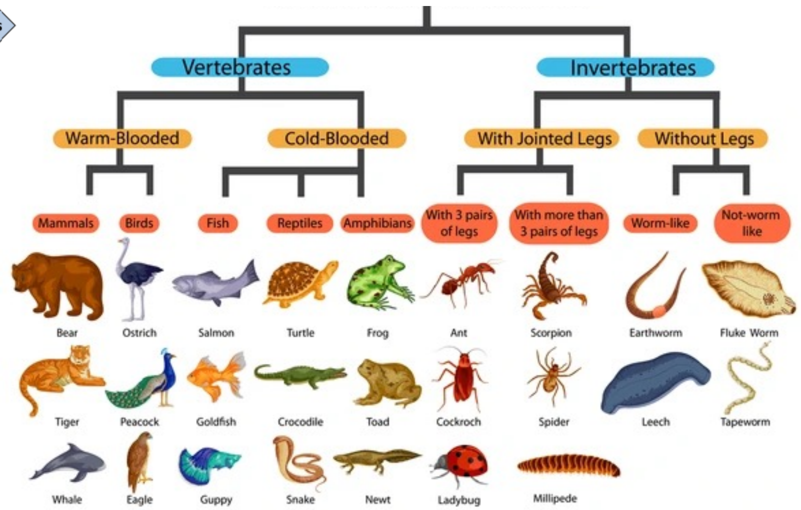
Description of processes



Structure of sentences

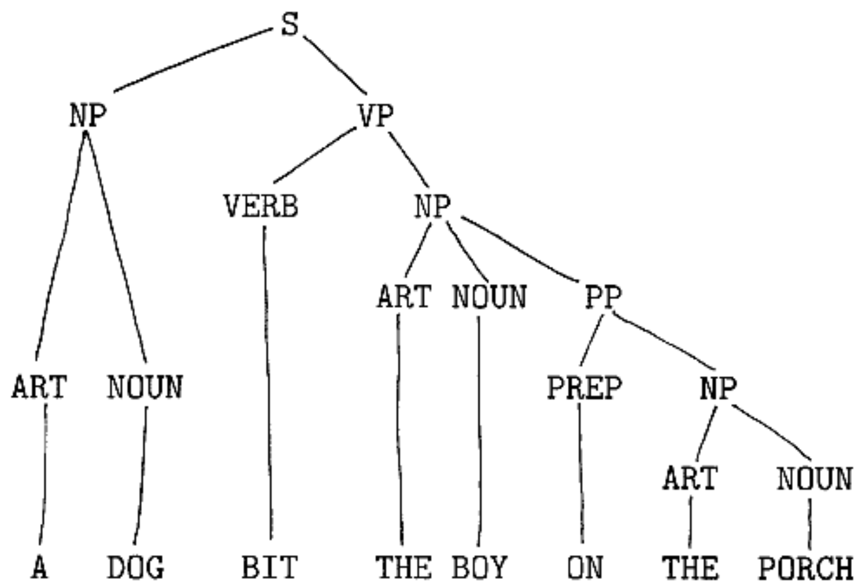


Classification of objects



Grammar

The **grammar** (art of letters) of a natural **language** is its set of **logical and structural rules** on speakers' or writers' usage and creation of clauses, phrases, and words.



γραμματικὴ τέχνη

Formal grammar

Formal grammar describes how to form strings from a language's alphabet that are valid according to the language's syntax.

A formal grammar is a **set of rules for rewriting strings**, along with a "start symbol" from which rewriting starts.

- **non-terminal** (phrases) and **terminal** (letters) symbols
- **initial non-terminal** symbol to start with
- **rewriting rules** in the form $u \rightarrow v$ (u and v are strings of symbols)

usage: $x\ u\ y \Rightarrow x\ v\ y$ *(substring is rewritten according to a rule)*

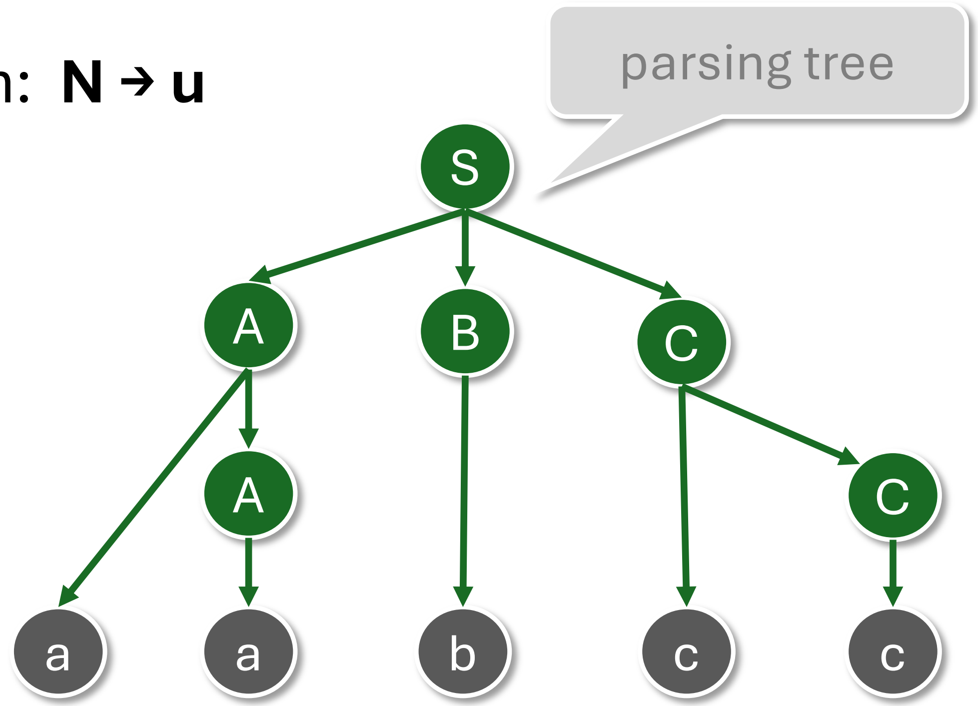
Context-free grammars

Rewriting rules have a specific form: **N → u**

Example:

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

$\{a^i b^j c^k\}$



Derivation of word:

S => **A**BC => a**A**BC => aa**B**C => aab**C** => aabc**C** => aabcc

Beyond context-free languages

What if we want the numbers of symbols a,b,c to be identical? The language is $\{a^n b^n c^n\}$ and this cannot be generated by CFG!

$S \rightarrow aSBC \mid aBC$

$CB \rightarrow BC$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

context-sensitive grammar

$S(n) \rightarrow A(k).B(l).C(m)$ $[n=k=l=m]$

$A(n) \rightarrow a$ $[n=1]$
$A(n) \rightarrow a.A(m)$ $[n=m+1]$

$B(n) \rightarrow b$ $[n=1]$
$B(n) \rightarrow b.B(m)$ $[n=m+1]$

$C(n) \rightarrow c$ $[n=1]$
$C(n) \rightarrow c.C(m)$ $[n=m+1]$

attribute grammar

Automated planning

Classical planning looks for a sequence of actions (*plan*) to achieve some *goal state*, where actions are connected via *causal relations* (action *effect* prepares *precondition* for later action).

..., go(Rob1,PosA,PosB), load(Rob1,Obj1,PosB),...

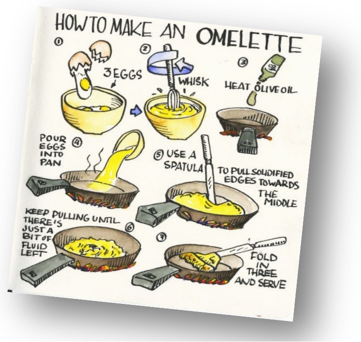
at(Rob1,PosB)

Hierarchical planning solves a goal task by decomposing it to sub-tasks until primitive tasks (actions) are obtained.

deliver(O,B) ⟶ **move**(R,A), load(R,O,A), **move**(R,B), unload(R,O,B)

move(R,Z) ⟶ [at(R,Z)]

move(R,Z) ⟶ go(R,X,Y), **move**(R,Z)



Decomposition method

$\mathbf{T \rightarrow T_1 \ldots T_k\ [C]}$

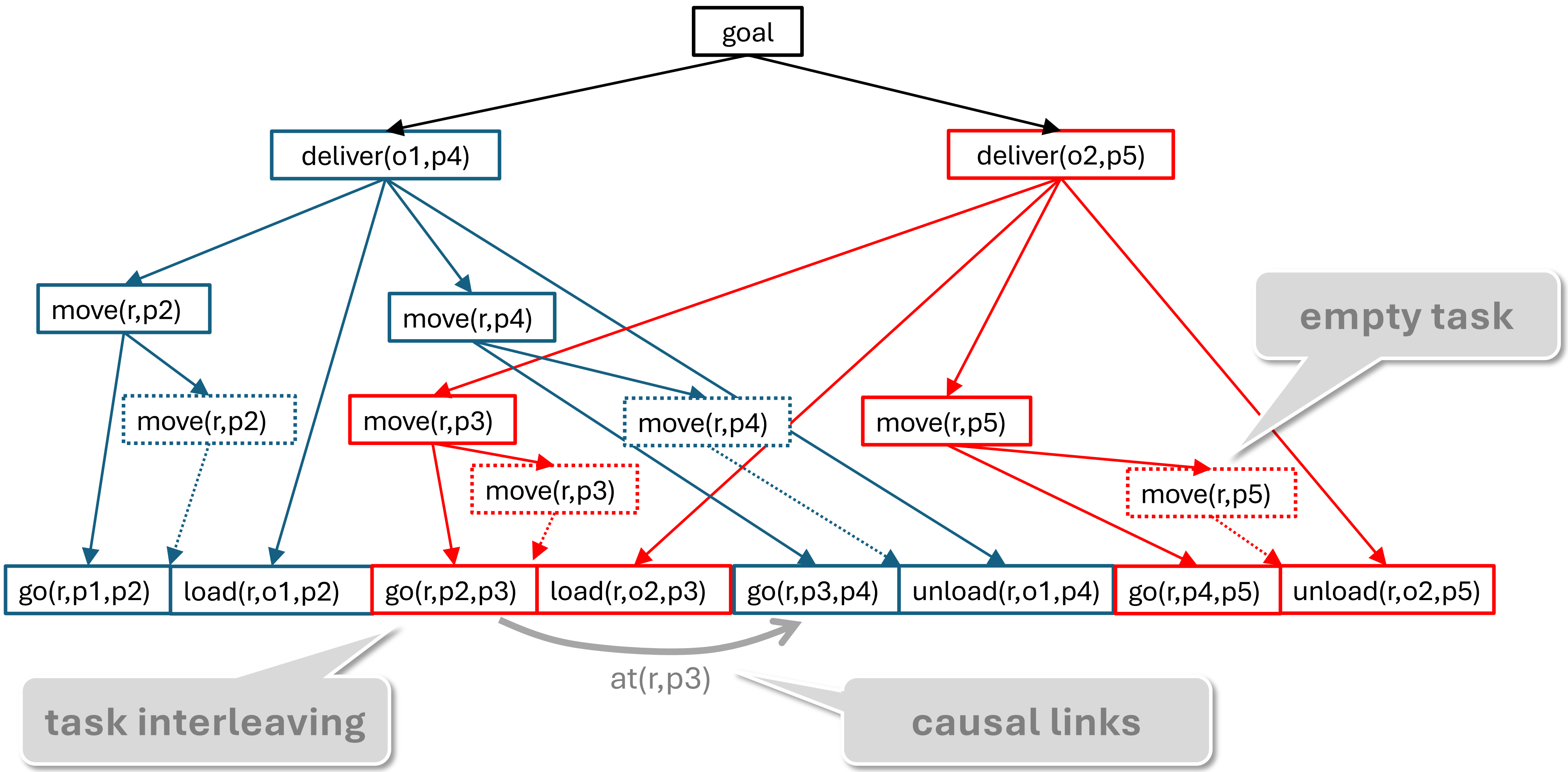
where **C** are decomposition constraints:

- $\mathbf{T_i < T_j}$: ordering of tasks
- **before(U,p)**: a precondition constraint (**p** is true right before the first task from **U**)
- **after(U,p)**: a postcondition constraint (**p** is true right after the last task from **U**), not an effect!!
- **between(U,p,V)**: prevailing constraints (**p** is true between the last task of **U** and the first task of **V**)

Note: the state constraints are defined between tasks but must be true between the actions obtained from the tasks

Holler, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2019. HDDL – A Language to Describe Hierarchical Planning Problems. arXiv:1911.05499.

Decomposition tree

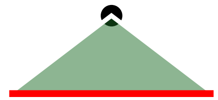


Planning Task

Given a goal task and initial state, find decomposition to an executable sequence of actions (a plan).

Why?

- to achieve a distant goal, agent needs a plan
- faster than classical planning
- better control over the generated plan



How?

- task decomposition and search
- SHOP2, PANDA, …

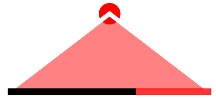
What if the task model is not enough to achieve a goal?

HTN with task insertion (**TIHTN**) allows inserting tasks/actions beyond the hierarchical structure

Nau, D. S.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. J. Artif. Intell. Res., 20: 379–404.

Plan (Goal) Recognition

Given a plan prefix (observed sequence of actions) and initial state, find a goal task (and missing future actions).



Why?

- deduce goals (and actions) of other agent(s) in cooperative and competitive environments

How?

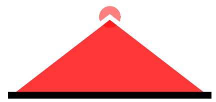
- via parsing (like in grammars)
- via planning (comparing plans to achieve various goals with observed actions)

Pantůčková, K.; and Barták, R. 2023. Using Earley Parser for Recognizing Totally Ordered Hierarchical Plans. In ECAI 2023, pp.1819-1826, IOS Press.

Plan Verification

Given a plan (action sequence) and initial state, check that the plan is valid:

- verify plan executability
- find a decomposition tree (and a goal task)



Why?

- verify that action sequence complies with the formal model

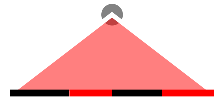
How?

- simulate action execution (plan executability)
- parsing (reconstruct a decomposition tree)

Barták, R.; Maillard, A.; and Cardoso, R. C. 2018. Validation of Hierarchical Plans via Parsing of Attribute Grammars. In Proc. of the 28th Int. Conf. on Automated Planning and Scheduling (ICAPS 2018), 11–19. AAAI Press.

Plan Correction

Given a plan (action sequence) and initial state, modify the plan to be valid with respect to hierarchical model.



Why?

- extension of plan verification for invalid plans
- plan recognition (correct a partially observed action sequence)
- planning („correct“ empty plan for a given goal task)

Ultimate planning-related technique

How?

- add and delete actions in the plan (via parsing)

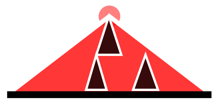
Barták, R.; Ondrčková, S.; Behnke, G.; and Bercher, P.: Correcting Hierarchical Plans by Action Deletion. In18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, 2021, 99–109.

Model Correction

Given a plan (action sequence) and initial state, modify the hierarchical model such that the plan is valid with respect to the model.

Why?

- automated construction of formal models from observations



Ultimate technique to bridge the knowledge acquisition gap.

How?

- machine learning approaches
- HTN Maker, HPNL, ...

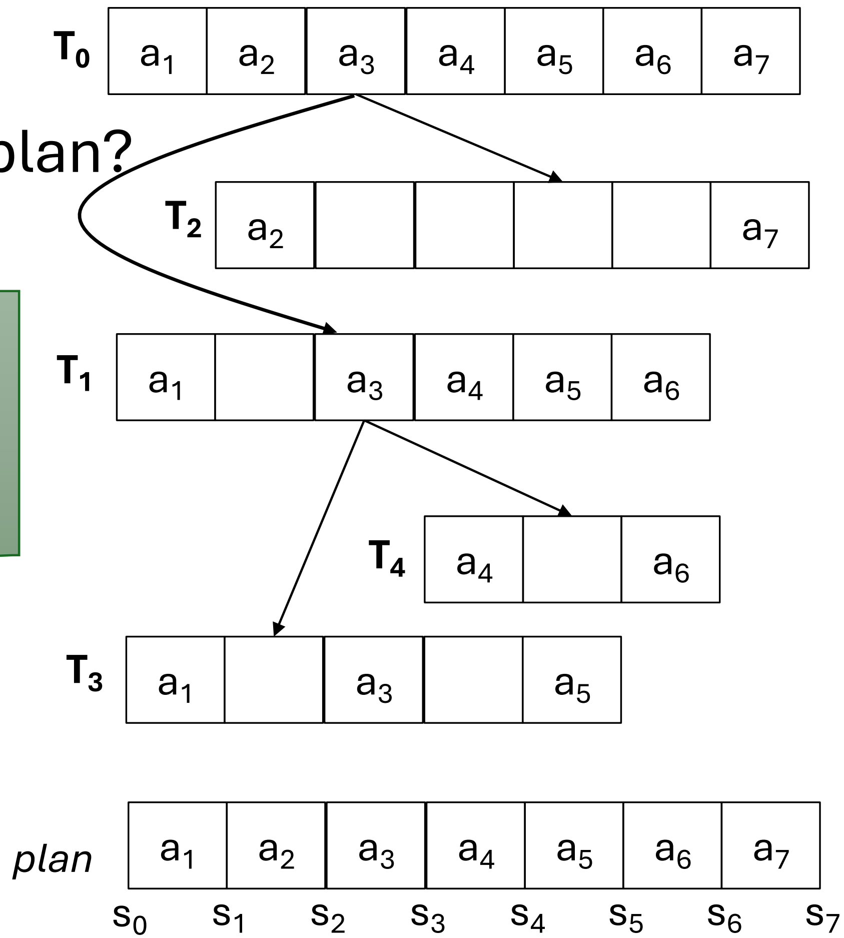
Hogg, C.; Munoz-Avila, H.; and Kuter, U. 2016. Learning Hierarchical Task Models from Input Traces. Comput. Intell., 32(1): 3–48.

Plan verification by parsing

Plan verification problem:
Given an action sequence, is it a valid HTN plan?

- causally consistent (executable)
- proper structure

$T_0 \dashrightarrow T_1\, T_2$
$T_1 \dashrightarrow T_3\, T_4$
$T_2 \dashrightarrow a_2\, a_7$
$T_3 \dashrightarrow a_1\, a_3\, a_5$
$T_4 \dashrightarrow a_4\, a_6$



Method:

1. Calculate all states
2. Verify plan executability
3. Group actions to tasks

Up to $O(T \times 2^N)$ pairs (task, subplan) generated
T – number of ground tasks
N – plan length

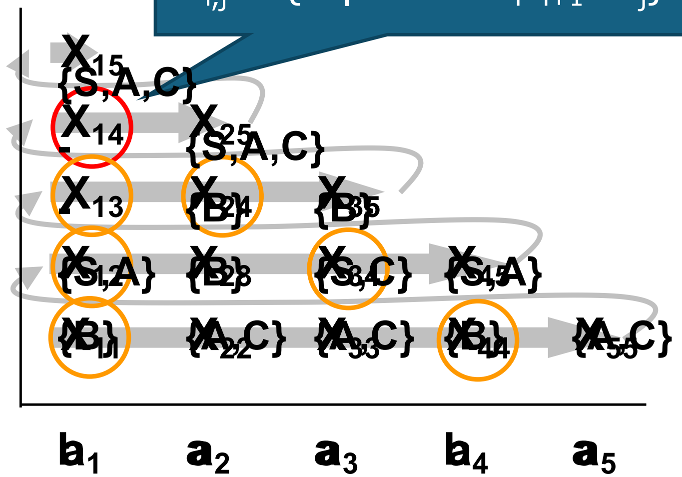
CYK-based Plan Verification

CYK (Cocke-Younger-Kasami) parsing

- **bottom-up** parsing for **context–free** grammars in **Chomsky Normal Form** exploiting **dynamic programming**

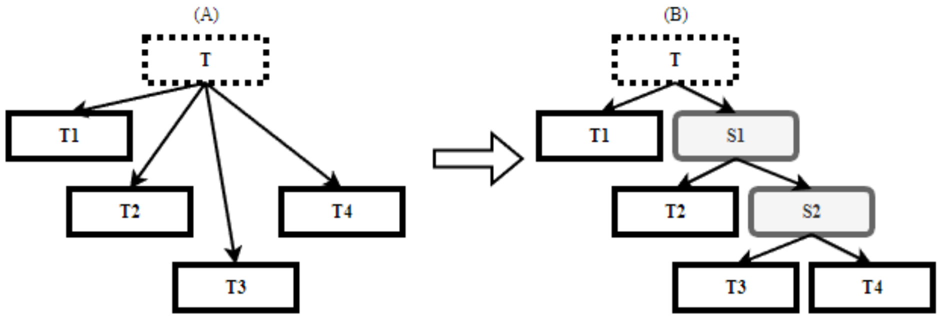
$X_{i,j} = \{A \mid A \Rightarrow^* a_i a_{i+1} \ldots a_j\}$

$S \to AB \mid BC$
$A \to BA \mid a$
$B \to CC \mid b$
$C \to AB \mid a$



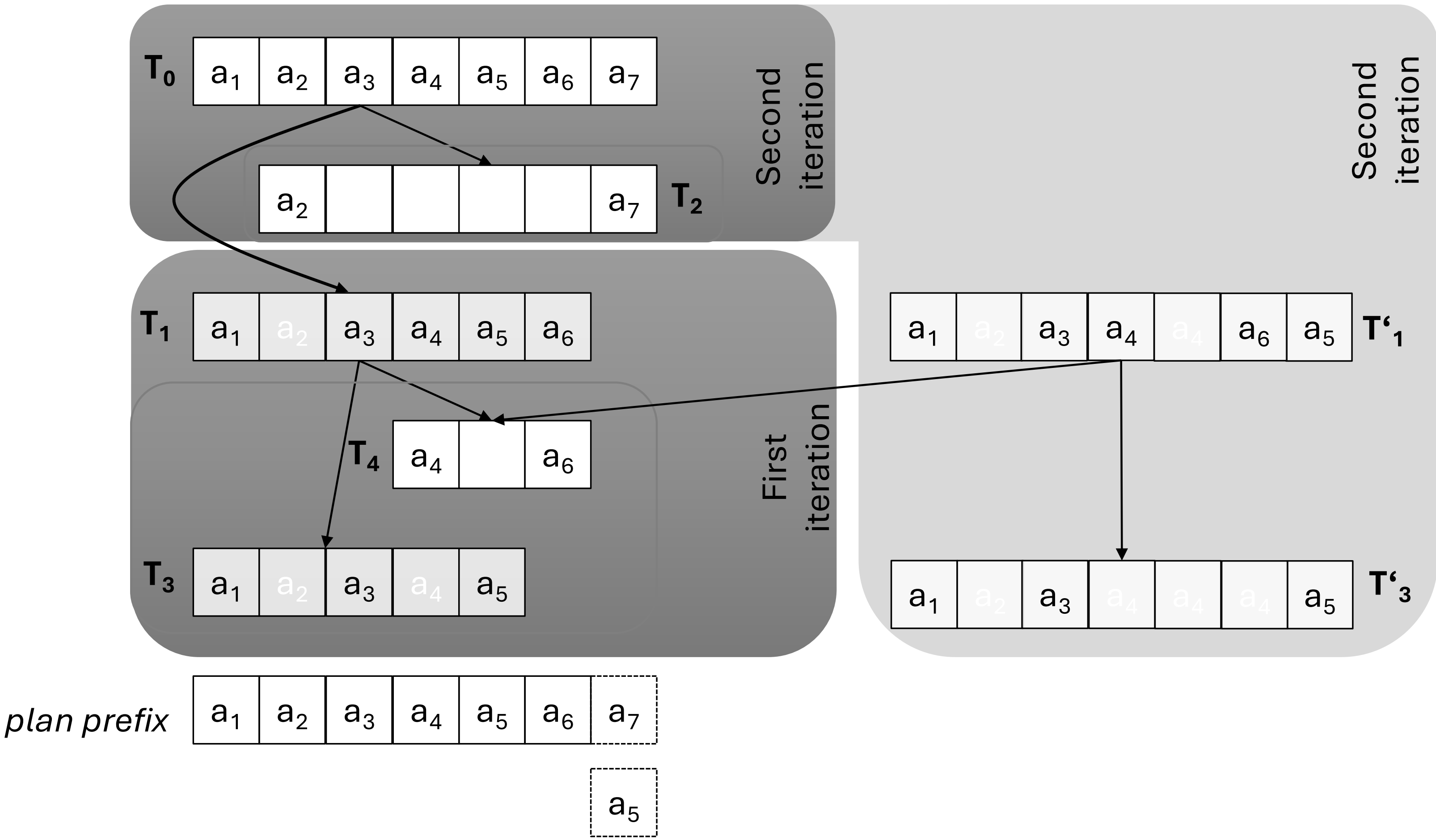
Application to **plan verification**:

- **2-regularization** (ChNF)
- **grounding**
- restriction to **totally-ordered domains**



Plan recognition by parsing

$T_0 \dashrightarrow T_1\, T_2$
$T_1 \dashrightarrow T_3\, T_4$
$T_2 \dashrightarrow a_2\, a_7$
$T_3 \dashrightarrow a_1\, a_3\, a_5$
$T_4 \dashrightarrow a_4\, a_6$



Earley parser

Developed for context-free grammars

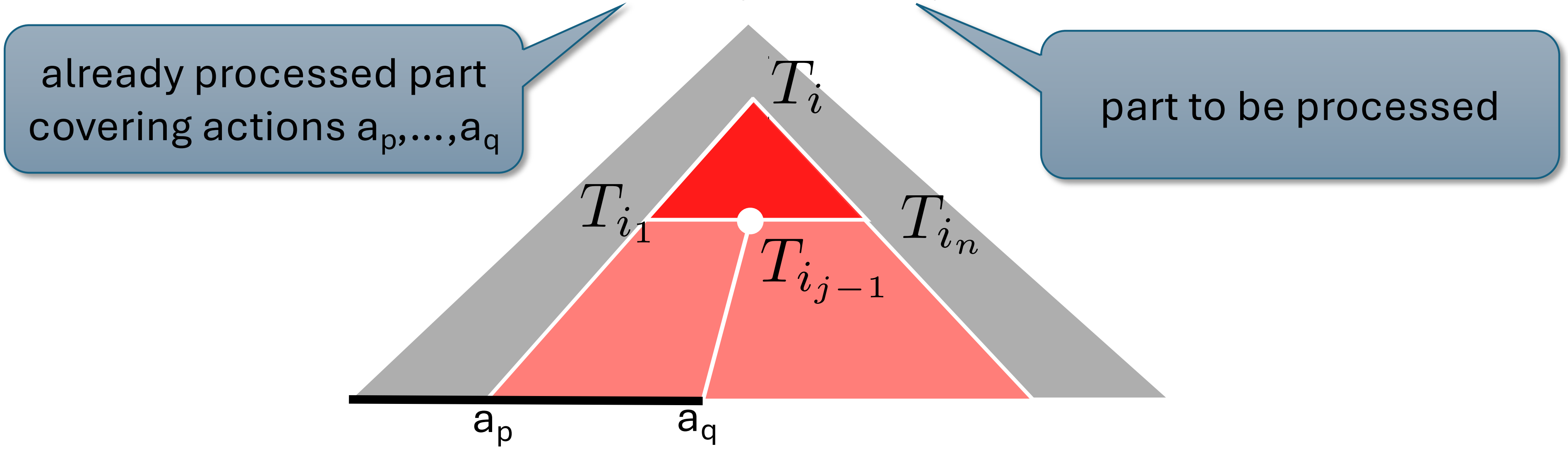
- restricted to totally-ordered HTN domains

Works top-down (from the root symbol towards the word)

- may support planning as well

Processes parsing states:

$$[T_i \rightarrow T_{i_1} \ldots T_{i_{j-1}} \bullet T_{i_j} \ldots T_{i_n}; p; q]$$



Earley parser: approach

Start with the **initial parsing state(s)**:

$$[G \rightarrow \bullet T; 0; 0]$$

Move the dot to the right as generating the decomposition (parsing) tree and scanning the input word by processing the parsing states:

- **predictor** (move down by applying a decomposition method)
- **scanner** (move right by reading an action from input plan)
- **completer** (move up by completing part of the tree)

Earley Parser: Predictor

$$[T_i \rightarrow T_{i_1} ... T_{i_{j-1}} \bullet T_{i_j} ... T_{i_n}; p; q]$$

If $\boldsymbol{T_{ij}}$ is a **compound task**, then

create parsing states corresponding to all decomposition methods for task $\boldsymbol{T_{ij}}$ (moving top-down)

$$[\ T_j \rightarrow \bullet T_{j_1} ... T_{j_m}; q; q\ ]$$

Earley Parser: Scanner

$$[T_i \rightarrow T_{i_1} ... T_{i_{j-1}} \bullet T_{i_j} ... T_{i_n}; p; q]$$

If $\boldsymbol{T_{ij}}$ is a **primitive task** (action) then

unify it with action ***a*** at position q+1 in the input plan

if unification is successful, then

create a new parsing state (moving left-to-right)

$$[\ T'_i \rightarrow T'_{i_1} ... T'_{i_{j-1}} a \bullet T'_{i_{j+1}} ... T'_{i_n}; p; q{+}1\ ]$$

Earley Parser: Completer

$$[\ T_i \rightarrow T_{i_1} ... T_{i_n} \bullet ; p ; q\ ]$$

If task ***T***

for each parsing state, where $\boldsymbol{T_i}$ unifies with $\boldsymbol{T_{jk}}$

$$[\ T_j \rightarrow T_{j_1} ... \bullet T_{j_k} ... T_{j_m} ; r ; p\ ]$$

create a new state (moving up)

$$[\ T'_j \rightarrow T'_{j_1} ... T'_{j_k} \bullet ... T'_{j_m} ; r ; q\ ]$$

Complete vision

Autonomous, adaptable agent with explainable and verifiable behavior

- Start with some **initial** hierarchical and action **model** (possibly empty)
- **Plan** for a given goal task, for example using TIHNT (i.e. using also classical planning)
- If the plan does not work, then re-plan, **repair the plan**
- If the plan is correct, **update the model** accordingly (learn new tasks and actions)
- In multi-agent environment
  - Use the model to predict actions of other agents
  - Update the model based on observations of other agents (learning by observation)