AAMAS’09, Budapest, 13.05.09

Operational Semantics of Goal Models in Adaptive Agents

Mirko Morandini[1], Loris Penserini[2] and Anna Perini[1]

[1]FBK IRST – Center for Information Technology, Trento, Italy

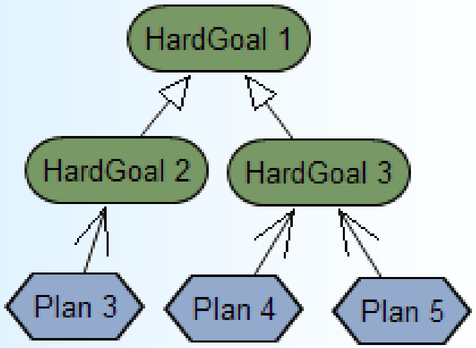
[2]Department of Computer Science, Universiteit Utrecht, NL

Outline

- Background and motivations
  - Goal models in software engineering
  - Goals in agent-oriented programming
  - Work objective
- Semantics for goal models at run-time
  - Semantics for “leaf”-goals [Riemsdijk08]
  - Semantics for goals in goal trees
- A small example
- Conclusions & future work

Goal models in Sw. Engineering

- from Goal-Oriented Requirements Engineering
  - Capture stakeholders' objectives
  - Analyse and structure them
  - Decompose goals,
    identify alternatives
  - Identify tasks (plans/capabilities)
    to perform, to achieve a goal



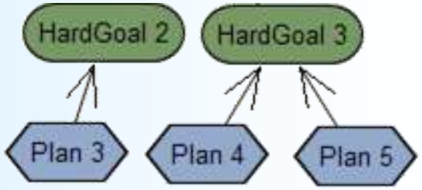
- used in KAOS, i*, many AOSE methodologies: Tropos, Prometheus, MaSE, Ingenias,… but most AOSE methodologies “loose” the concept of goal in the later development phases!

Research question:

How to use this knowledge to **shift decision making** (evaluation of alternatives) **from design- to run-time**, to gain in autonomy, for the development of adaptive and fault-tolerant systems?

Goals in agent-oriented programming

- Jason, 2APL, Jadex, Jack:
  - BDI-architecture: Goals, Plans, Beliefs
  - Represent "operationalised" goals, with possible plans to achieve them (goal model *"leaf level"*).



  - Plans can contain activities to execute and other goals to achieve.
  - Various goal types for a specific run-time behaviour (achieve, maintain, perform,…) [Dastani06]

Research question:

How can we deal with goal models at run-time?

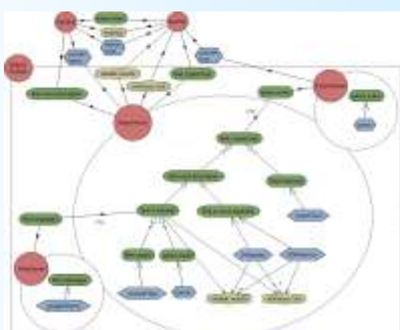
From goal models to run-time

Maintain goal models also at implementation and run-time!

Previous work

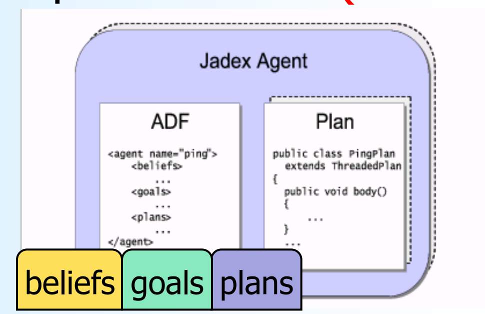
- Tropos4AS: extends the AOSE methodology **TROPOS** for modelling properties of adaptive systems [Morandini08]:
  - goal types
  - conditions to the environment
- t2x: automated mapping of Tropos4AS goal models to Jadex BDI agents [PenseriniAAMAS07]

Agent-Oriented Design (TROPOS)



t2x tool

BDI Agent-Oriented Implementation (Jadex)



Work Objective

- Goal models in most AOSE methodologies, but “lost” in the later development phases
- Agent languages: goals, but no support for goal structures
- We have an (informal) mapping of goal models to code

Try to formalise the intended behaviour of the satisfaction process for a goal model!

Goal models at run-time – motivation:

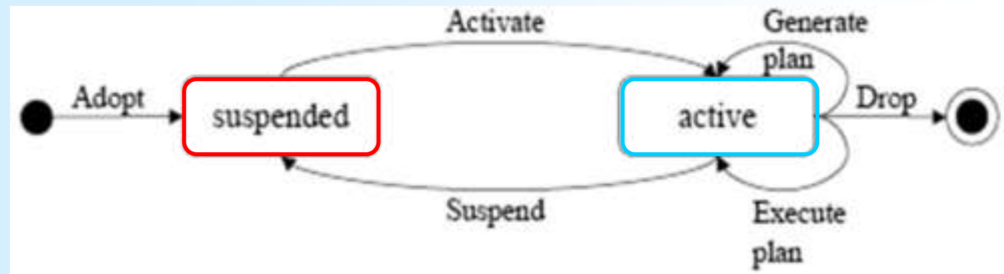
- Maintain high-level design information and traceability of the requirements
- Use this knowledge to shift design decisions (evaluation of alternatives) to run-time to gain in autonomy, for the development of adaptive and fault-tolerant systems

Semantics for leaf goals [Riemsdijk08]

B. van Riemsdijk, M. Dastani and M. Winikoff, "Goals in Agent Systems: An Unifying Framework", AAMAS, 2008.

Unified representation of operational semantics for the different goal types available in current agent programming languages.

- Abstract architecture for goals
  - possible goal states
  - operational semantics defined by transition rules



e.g.

Activation condition c true on current belief

$\langle$belief, goal susp.$\rangle \rightarrow \langle$belief, goal activated$\rangle$

$$\frac{\langle c, \text{ACTIVATE} \rangle \in E \quad B \models c}{\langle B, \mathsf{g}(C, E, \text{SUSPENDED}, \epsilon)\rangle \rightarrow \langle B, \mathsf{g}(C, E, \text{ACTIVE}, \epsilon)\rangle}$$

Formalisation of common goal types

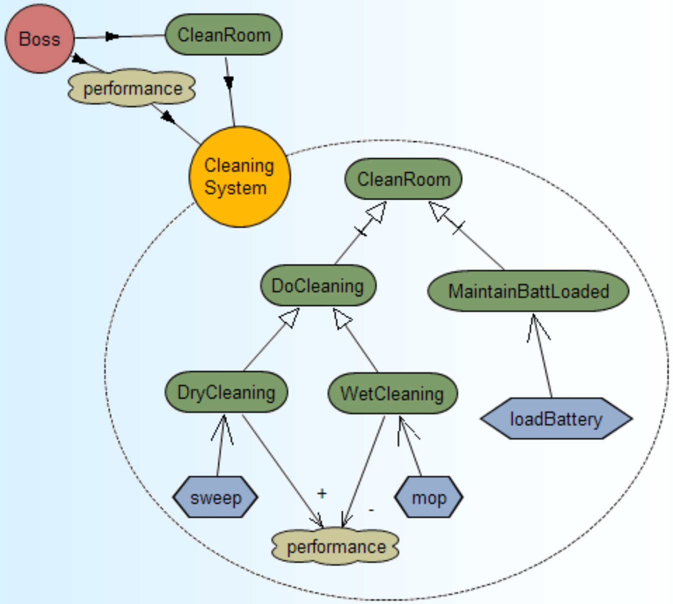
e.g.: "Achieve-goal" with satisfaction condition s and failure condition f :

$$\mathsf{A}'(s, f) \equiv \mathsf{g}(\{\}, \{\langle s \vee f, \text{DROP}\rangle, \langle true, \text{ACTIVATE}\rangle\})$$

Semantics for non-leaf goals

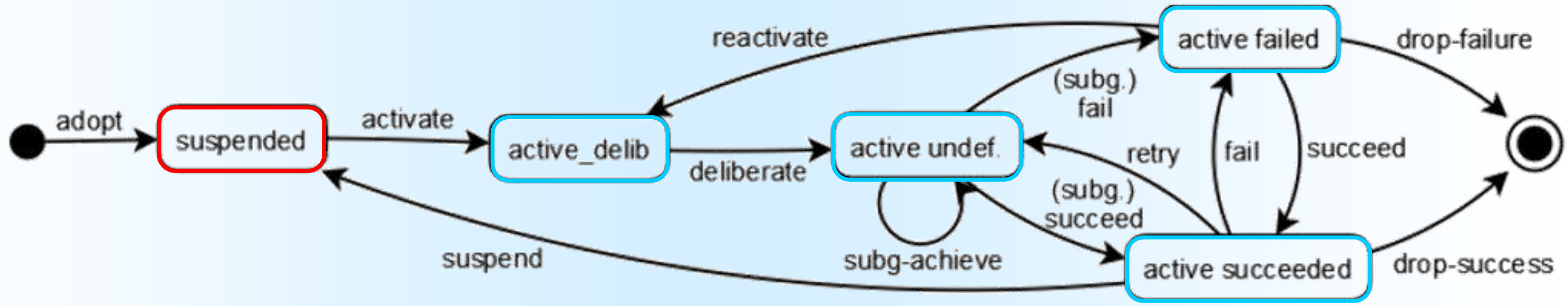
Challenges:

- Semantics for goal AND-OR decompositions,
- Interplay between subgoal satisfaction and the satisfaction of the achievement conditions for different goal types,
- Customisable formalisation to capture different satisfaction behaviours.



Semantics for non-leaf goals

Extend [Riemsdijk08] for non-leaf goals in goal models



"Active" state extended to

- "Active, deliberate" (AD): get applicable subgoals
- "Active, undefined" (AU): subgoal achievement taking place, result still undefined
- "Active, succeeded" (AS): "provisional" success state. Subgoal achievement succeeded, evaluate goal achievement conditions
- "Active, failed" (AF): "provisional" failure state. Subgoal achievement failed, evaluate goal achievement conditions

Transition rules – example for OR:

In state AU, try to achieve a subgoal, if it succeeds, go to AS

$$\frac{\gamma_i \in \Gamma \quad \langle B, adopt(G,\gamma_i)\rangle \to \langle B', G\rangle \quad B' \models success(\gamma_i)}{\langle B, g(C,E,AU,\Gamma)\rangle \to \langle B', g(C,E,AS,\Gamma \setminus \{\gamma_i\})\rangle}\;[OR{:}subg\text{-}succeed]$$

Instantiation of the abstract architecture for different goal types

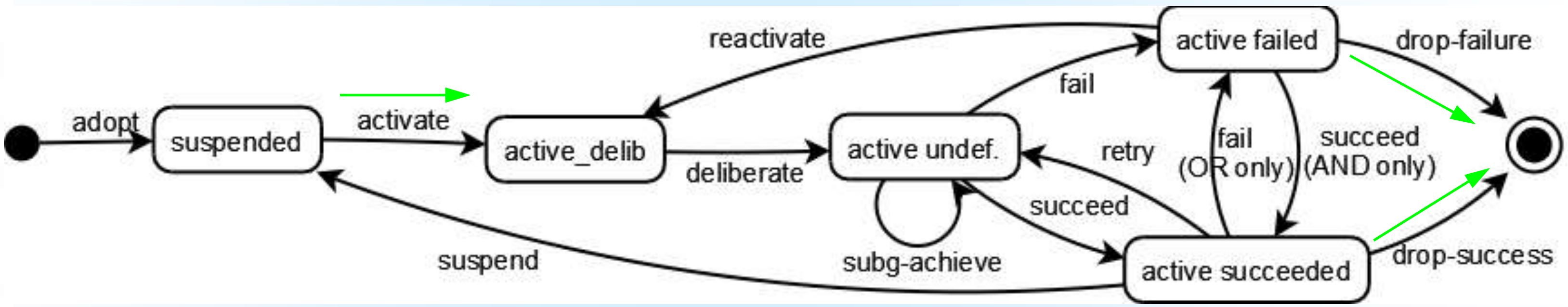
The behaviour of the different goal types can be defined by defining the conditions linked to the transition actions.

[E: conditions evaluated when the list of subgoals to achieve is empty]

[C: conditions evaluated when the list of subgoals is not empty]

$$P \equiv g(E,C), with \;\; E = C = \{\langle true, \text{ACTIVATE}\rangle, \langle true, \text{DROPFAILURE}\rangle, \langle true, \text{DROPSUCCESS}\rangle\}$$

Perform-Goal



Achieve-Goal

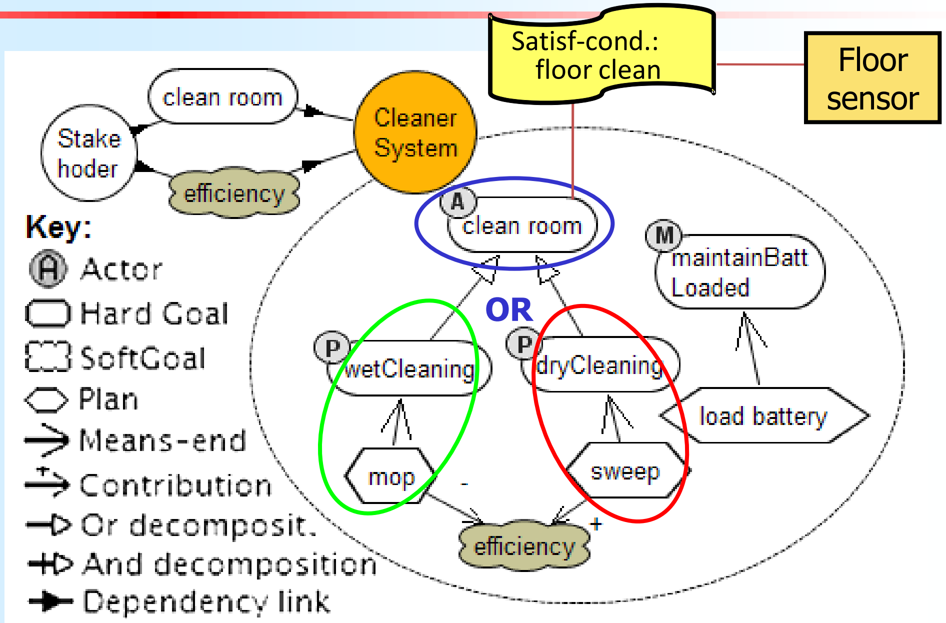
success & failure conditions

$$A(s,f) \equiv g(E,C), \;\; with \; E = H \cup \{\langle \neg s \vee f, \text{FAIL}\rangle\}$$
$$and \; C = H \cup \{\langle f, \text{FAIL}\rangle, \langle \neg s, \text{RETRY}\rangle\}$$
$$H = \{\langle true, \text{ACTIVATE}\rangle, \langle f, \text{DROPFAILURE}\rangle, \langle s, \text{SUCCEED}\rangle, \langle s, \text{DROPSUCCESS}\rangle, \langle \neg s, \text{REACTIVATE}\rangle\}$$

A small example

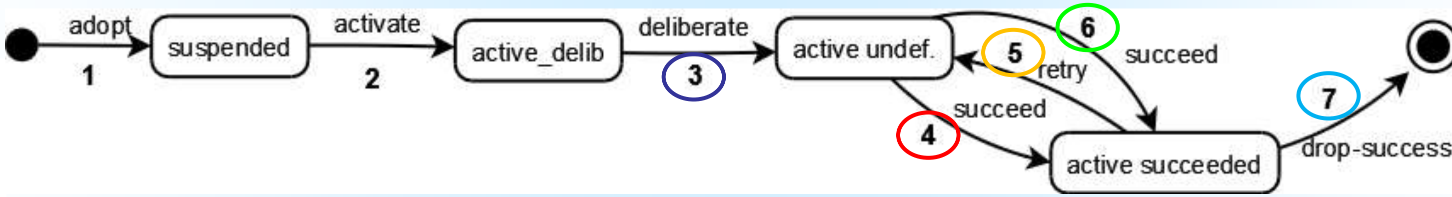
Cleaner Robot:

Should clean a room, with satisfaction condition "floor clean".

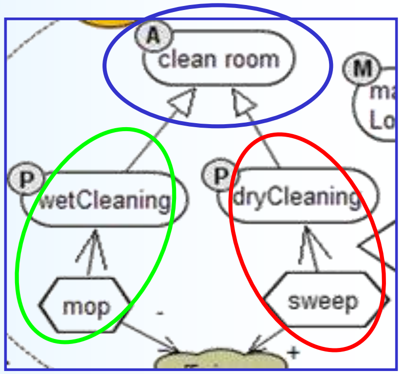


A scenario:

- Robot cleans the floor, achieving "dryCleaning".
- Sweeping performed, still some dirt spots on the floor! The agent tries "wetCleaning".
- Cleaning fails, because it runs out of water,
  → but dirty area already cleaned,
  → top goal "clean room" achieved with success!



Satisfy the achieve-goal ***clean room***



- 3) deliberate(g, B) gives back subgoals *wetCleaning* (WC) and *dryCleaning* (DC)

$$\frac{}{\langle B, g(C,E,AD,\emptyset)\rangle \rightarrow \langle B, g(C,E,AU, deliberate(g,B))\rangle} \; [deliberateE]$$

- 4) *dryCleaning* performed with success

$$\frac{\dfrac{disp(G,DC) \rightarrow G \cup \{DC\} \qquad \langle B, G \cup \{DC\}\rangle \rightarrow \langle B', G\rangle}{\langle B, disp(G,DC)\rangle \rightarrow \langle B', G\rangle} \qquad B' \models success(DC)}{\langle B, g(C,E,AU,\{DC,WC\})\rangle \rightarrow \langle B', g(C,E,AS,\{WC\})\rangle}$$

- 5) still some dirt spots on the floor! only $\langle \neg s, \textsc{Retry}\rangle$ is satisfied.

$$\frac{\Gamma \neq \emptyset \qquad \langle c, \textsc{Retry}\rangle \in C \qquad B \models c}{\langle B, g(C,E,AS,\Gamma)\rangle \rightarrow \langle B, g(C,E,AU,\Gamma)\rangle} \; [Retry]$$

- 6) *wetCl*. is pursued and fails, but condition *"floor clean"* is now true.

$$\frac{\neg\exists\langle d, \textsc{Fail}\rangle \in C.(B \models d) \qquad \langle c, \textsc{Succeed}\rangle \in E \qquad B \models c}{\langle B, g(C,E,AU,\emptyset)\rangle \rightarrow \langle B, g(C,E,AS,\emptyset)\rangle} \; [cond\text{-}succeedE]$$

$$\frac{g(C,E,AS,\emptyset) \in G \qquad \langle c, \textsc{DropSuccess}\rangle \in E \qquad B \models c}{\langle B, G\rangle \rightarrow \langle B \cup \underline{success(g)}, G \setminus \{g(C,E,AS,\emptyset)\}\rangle} \; [drop\text{-}successE]$$

7) Finally the goal is dropped and its success is annotated in the belief base.

Conclusions & Future Work

- We formalised the run-time behaviour of non-leaf goals, defining the interplay between goal decompositions and goal types.
- The proposed ‘abstract architecture’ can be used to define various goal types and achievement/failure handling behaviours.
- Maintain high-level design information and traceability of the requirements
- shift decisions (evaluation of alternatives) from design to run-time to gain in autonomy, for the development of adaptive and fault-tolerant systems
- The operational semantics can be a starting point:
  - to formalise a mapping from goal models to software agents,
  - to implement a middle layer for goal models in AOP frameworks,
  - for validation and simulation of goal models at design time.
- Goal models at run-time also provide a basis for run-time goal acquisition and goal model modification.

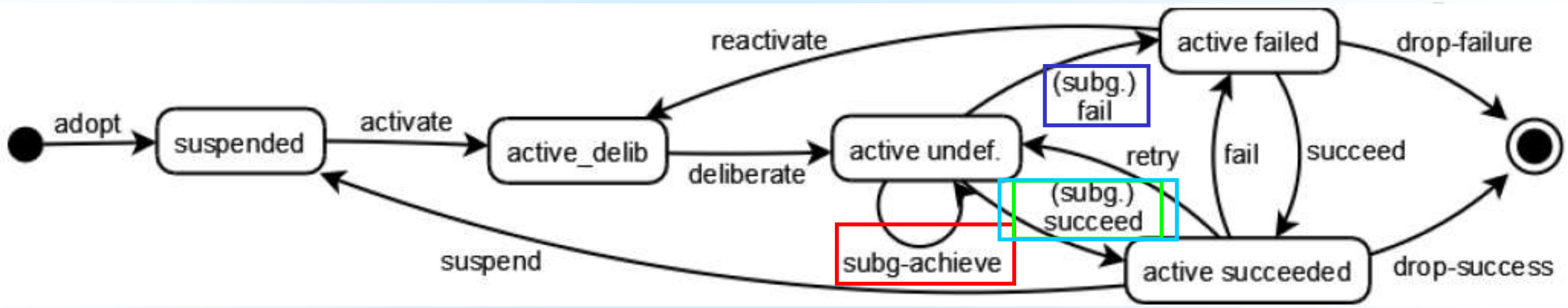
Thank you!

Questions and suggestions are welcome!

Further readings & references

- [Bresciani04] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems, 8(3):203–236, July 2004.
- [Penserini07] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. High variability design for software agents: Extending Tropos. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2(4), 2007.
- [PenseriniAAMAS07] L. Penserini, A. Perini, A. Susi, M. Morandini, and J. Mylopoulos, A design framework for generating BDI agents from goal models. AAMAS'07, Honolulu, 2007.
- [Morandini ASE08] M. Morandini, L. Penserini, and A. Perini. Automated mapping from goal models to self-adaptive systems. ASE'08, L'Aquila, Italy, September 2008.
- [SEAMS08] M. Morandini, L. Penserini, and A. Perini. Towards Goal-Oriented Development of Self-Adaptive Systems. SEAMS at ICSE08, Leipzig, Germany, May 2008.

- [Riemsdijk08] B. van Riemsdijk, M. Dastani and M. Winikoff. Goals in Agent Systems: An Unifying Framework. AAMAS'08, Estoril, Portugal, May 2008.
- [Dastani06] M. Dastani et al., Goal types in agent programming. AAMAS06, 2006.
- [Pokahr05] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A bdi reasoning engine. In *Multi-Agent Programming*, pages 149–174, 2005. Book chapter.
- [Yu95] E. Yu. Modelling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto, Department of Computer Science, University of Toronto, 1995.



- Some transitions guided by transition actions (**Succeed, Fail, Retry,…**) linked to a condition *c*, evaluated on the agent's belief B.

Example transition rules for OR-decomposition

• in state AU: try to achieve a subgoal, if it fails, remain in AU.

$$\frac{\gamma_i \in \Gamma \quad \langle B, adopt(G,\gamma_i)\rangle \rightarrow \langle B', G\rangle \quad B' \models failure(\gamma_i)}{\langle B, g(C,E,AU,\Gamma)\rangle \rightarrow \langle B', g(C,E,AU,\Gamma \setminus \{\gamma_i\})\rangle} \; [OR{:}subg\text{-}achieve]$$

• in state AU, try to achieve a subgoal, if it succeeds, go to AS

$$\frac{\gamma_i \in \Gamma \quad \langle B, adopt(G,\gamma_i)\rangle \rightarrow \langle B', G\rangle \quad B' \models success(\gamma_i)}{\langle B, g(C,E,AU,\Gamma)\rangle \rightarrow \langle B', g(C,E,AS,\Gamma \setminus \{\gamma_i\})\rangle} \; [OR{:}subg\text{-}succeed]$$

• in AU or AF, if success condition ***c*** is true and failure condition ***d*** false, go to AS

$$\frac{\Gamma \neq \emptyset \qquad \neg\exists\langle d, \textsc{Fail}\rangle \in C.(B \models d) \quad \langle c, \textsc{Succeed}\rangle \in C \quad B \models c}{\langle B, g(C,E,\mathbf{X},\Gamma)\rangle \rightarrow \langle B, g(C,E,AS,\Gamma)\rangle \qquad \mathbf{X} \in \{AU, AF\}} \; [cond\text{-}succeedC]$$

• in AU, if no more subgoals to achieve and success condition true, go to AF.

$$\frac{\neg\exists\langle c, \textsc{Succeed}\rangle \in E.(B \models c)}{\langle B, g(C,E,AU,\emptyset)\rangle \rightarrow \langle B, g(C,E,AF,\emptyset)\rangle} \; [OR{:}subg\text{-}fail]$$