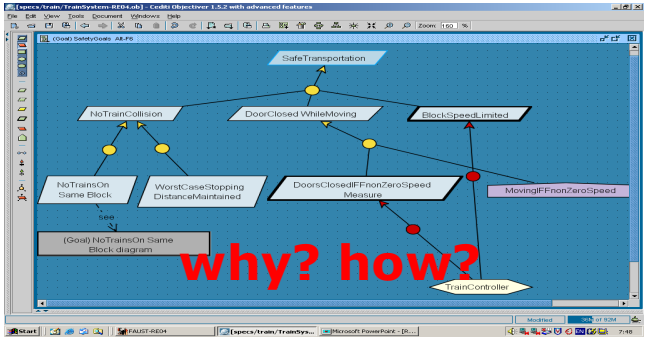
Modeling System Objectives with Goal Diagrams

Mariano Ceccato

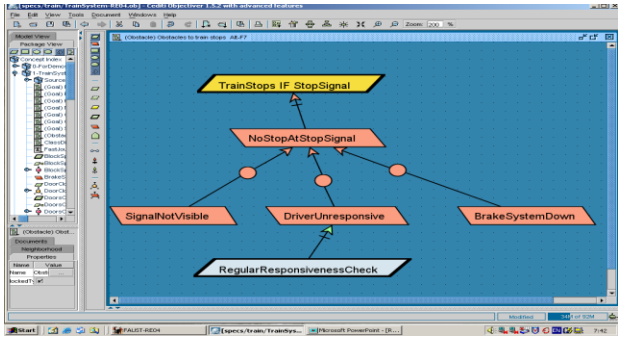
mariano.ceccato@univr.it

Building models for RE

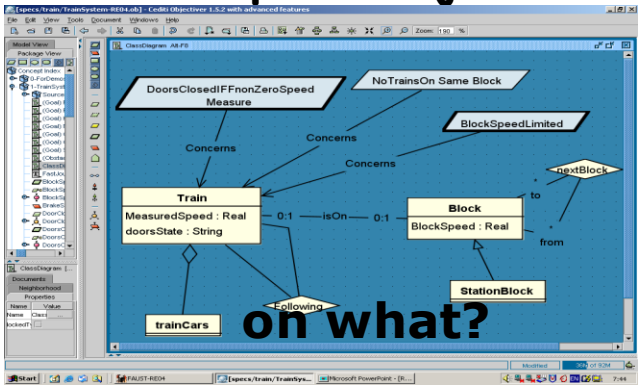
Goals



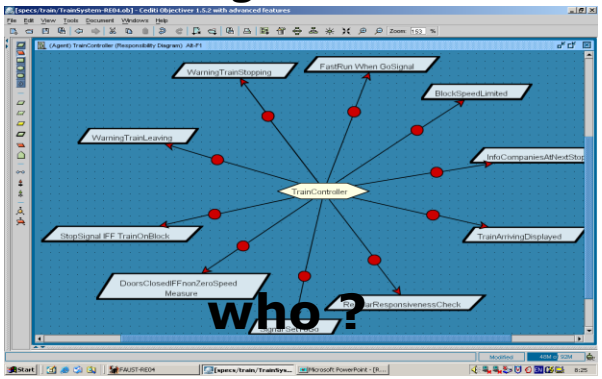
Risks



Conceptual objects



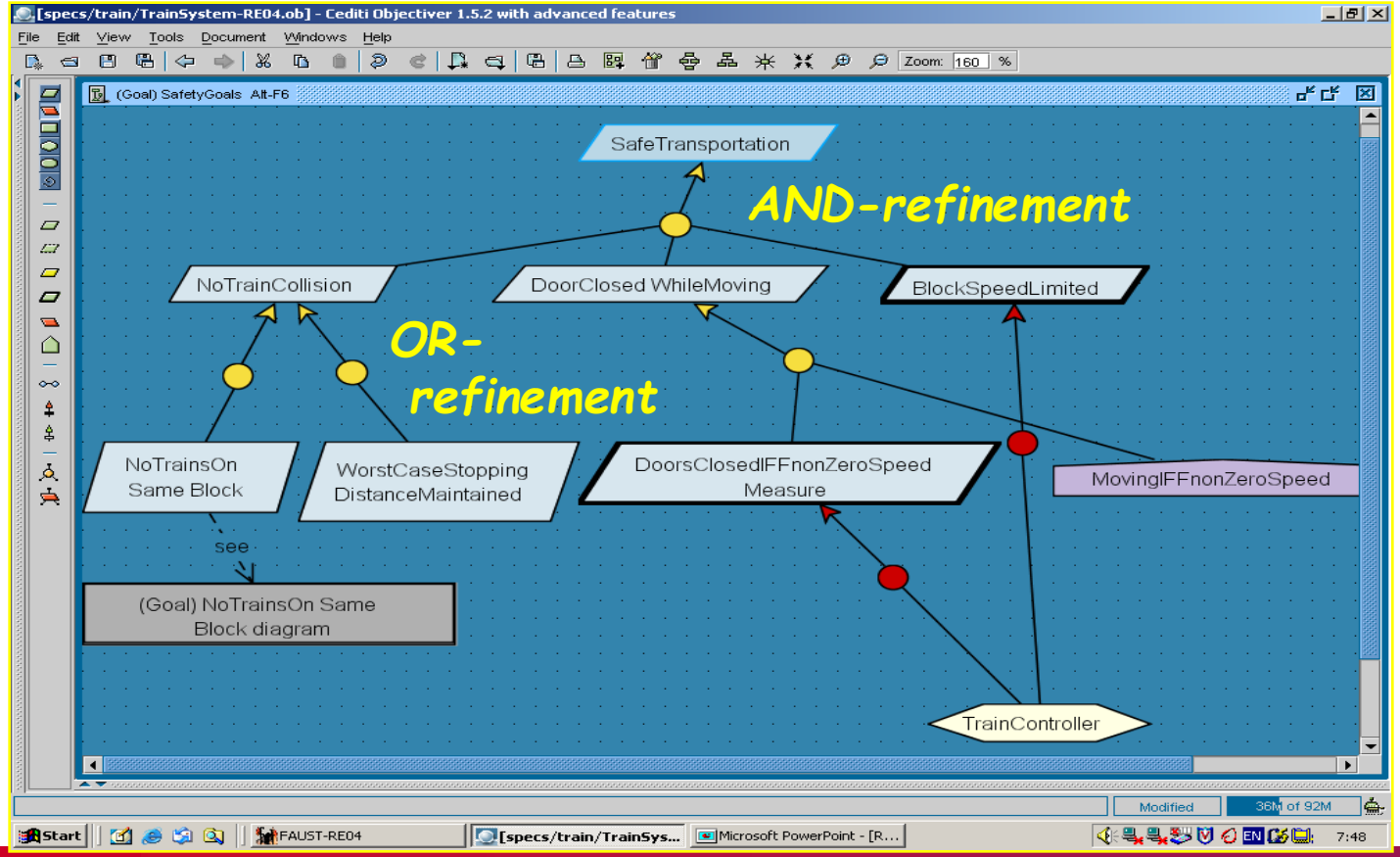
Agents



Goals as seen in “Goal orientation” lecture

- Prescriptive statements of intent the system should satisfy through cooperation of its agents
  - formulated in terms of problem world phenomena
  - at various levels of abstraction/granularity
- Can be negotiated, weakened, prioritized (unlike domain props)
- The finer-grained a goal, the fewer agents required for its satisfaction
  - requirements, expectations: single-agent goals
- Behavioral (Achieve/Maintain) goals, soft goals
- Functional, quality, development goals

A goal model shows contribution links and leafgoal assignments



Goal modeling: outline

- Goal features as model annotations
- Goal refinement
- Capturing conflicts among goals
- Connecting the goal model with other system views
- Capturing alternative options
- Goal diagrams as AND/OR graphs
- Documenting goal refinements & assignments with annotations
- Building goal models: heuristic rules & reusable patterns

Goal features are specified in model annotations

DoorsClosedWhileMoving

goal

annotation

Goal *Maintain* [DoorsClosedWhileMoving]

Def *All train doors shall be kept closed at any time when the train is moving*

[FormalSpec ... in temporal logic for analysis]

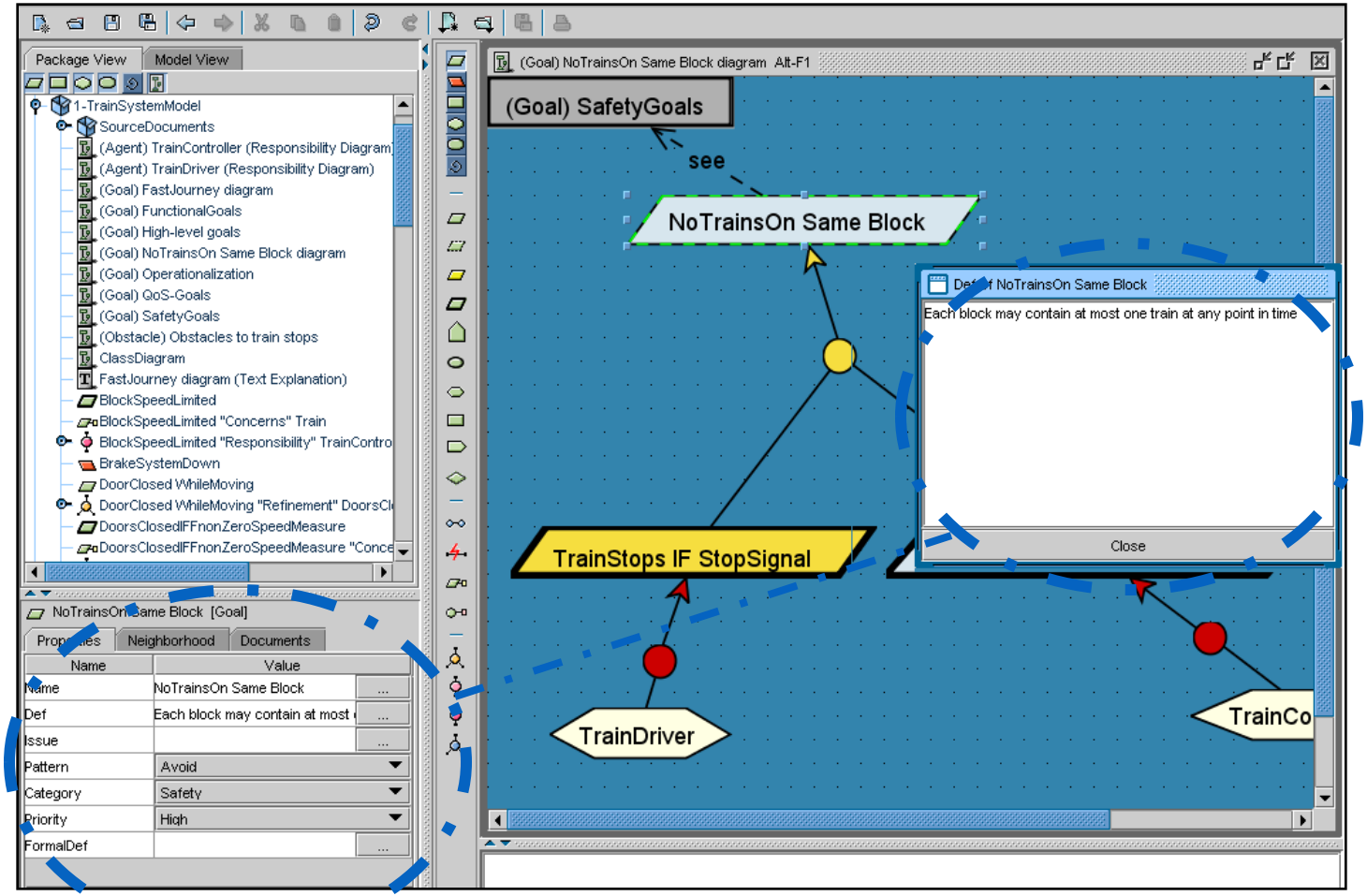
[Category Safety]

[Priority Highest]

[Source From interview with railway engineer X ...]

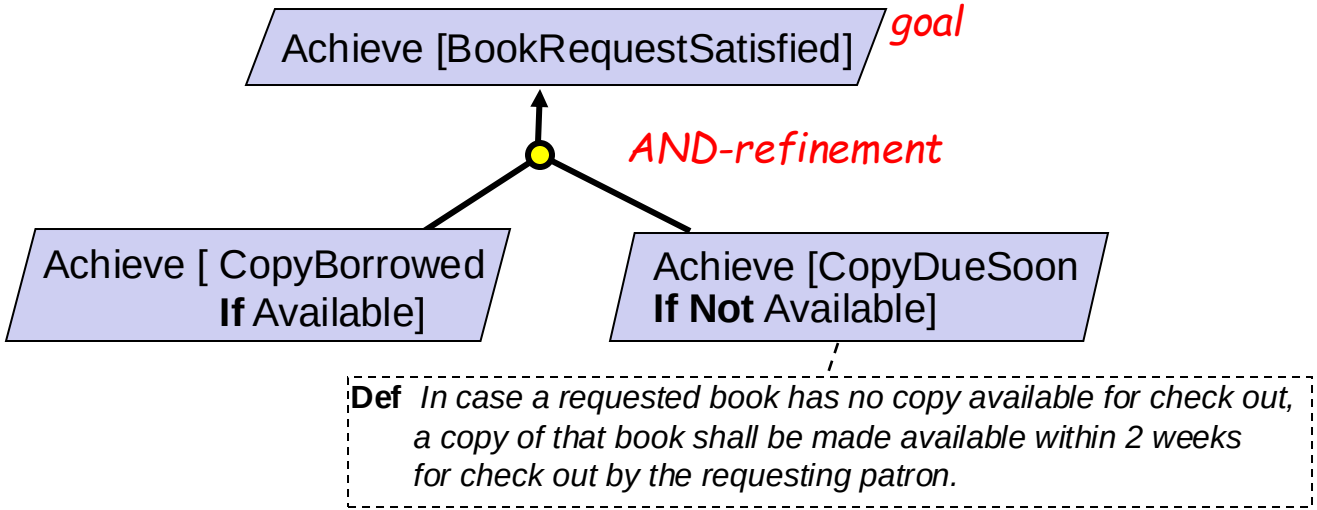
precise definition

features


Package View
Model View
1-TrainSystemModel
SourceDocuments
(Agent) TrainController (Responsibility Diagram)
(Agent) TrainDriver (Responsibility Diagram)
(Goal) FastJourney diagram
(Goal) FunctionalGoals
(Goal) High-level goals
(Goal) NoTrainsOn Same Block diagram
(Goal) Operationalization
(Goal) QoS-Goals
(Goal) SafetyGoals
(Obstacle) Obstacles to train stops
ClassDiagram
FastJourney diagram (Text Explanation)
BlockSpeedLimited
BlockSpeedLimited "Concerns" Train
BlockSpeedLimited "Responsibility" TrainContro
BrakeSystemDown
DoorClosed WhileMoving
DoorClosed WhileMoving "Refinement" DoorsCl
DoorsClosedIFFnonZeroSpeedMeasure
DoorsClosedIFFnonZeroSpeedMeasure "Conce
NoTrainsOn Same Block [Goal]
Neighborhood
Documents
Name
Value
NoTrainsOn Same Block
Each block may contain at most
Pattern
Avoid
Category
Safety
Priority
High
FormalDef
(Goal) NoTrainsOn Same Block diagram Alt-F1
(Goal) SafetyGoals
see
NoTrainsOn Same Block
Each block may contain at most one train at any point in time
Close
TrainStops IF StopSignal
TrainDriver
TrainCo

Goal refinement

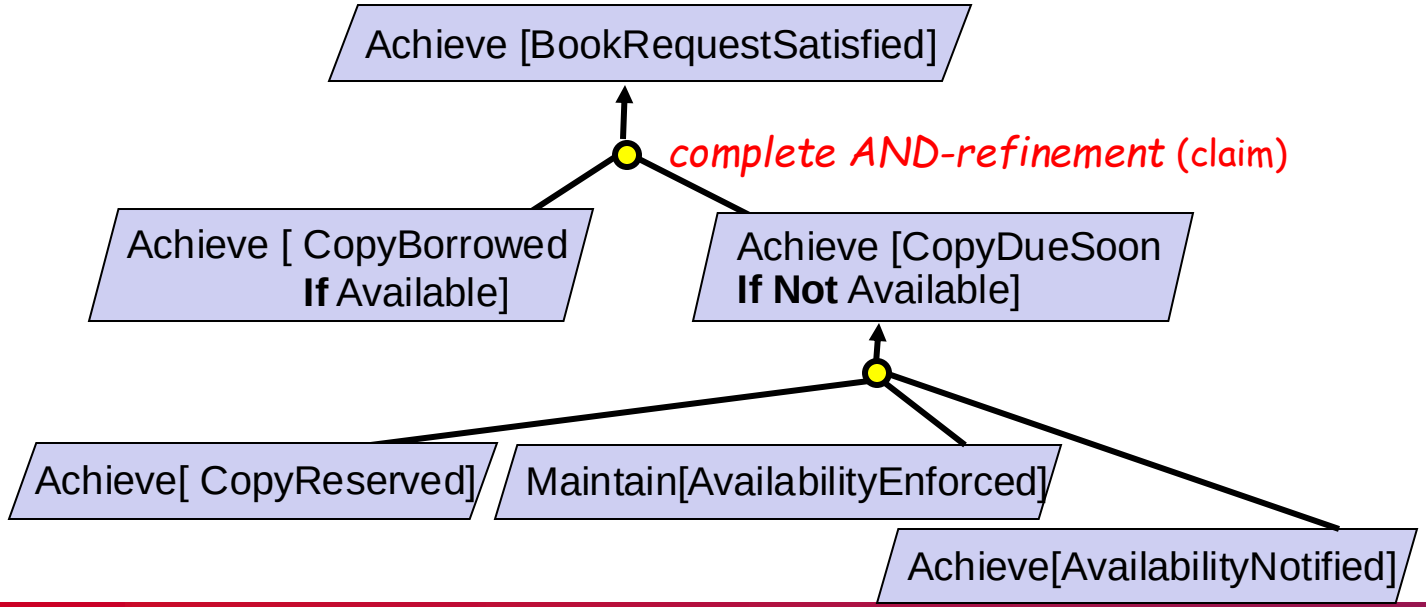
- An AND-refinement of goal G into subgoals G_1, ..., G_n states that G can be satisfied by satisfying G_1, ..., G_n
  - The set {G_1, ..., G_n} is called refinement of G
  - Subgoal G_i is said to contribute positively to G



AND-refinements should be complete

- $\{G_1, ..., G_n\}$ is a **complete AND-refinement** of G iff satisfying $G_1, ..., G_n$ is sufficient for satisfying G in view of known domain properties

  $\{G_1, ..., G_n, Dom\} \models G$



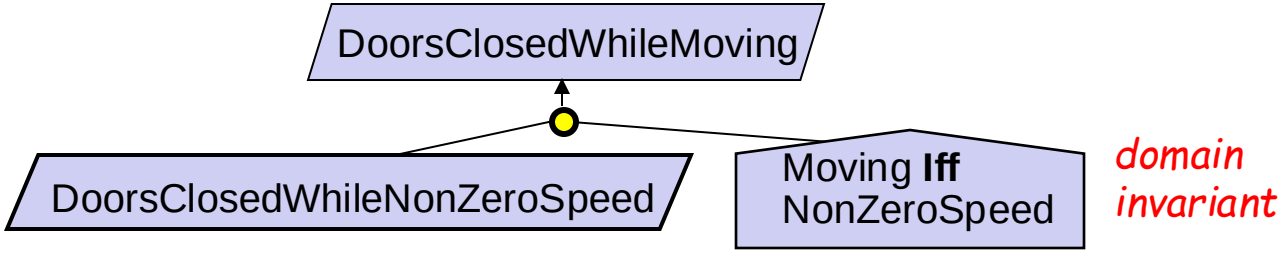
Complete AND-refinements

- Getting complete refinements of behavioral goals is essential for **requirements completeness**
- Domain properties are often used for arguing about complete refinements
  - classified as
    - domain invariants: known to hold in every state

      "train doors are either open or closed"
    - domain hypotheses: assumed to hold in specific states

      "railway tracks are in good conditions ..."
  - attached to conceptual objects in the object model

Domain properties in AND-refinements



AND-refinements should also be consistent and minimal

- **Consistent**: subgoals $G_1, \ldots, G_n$ and domain properties in *Dom* may not contradict each other:

$$\{G_1, \ldots, G_n, \text{Dom}\} \not\models \textbf{false}$$

(any behavior would be permitted from **false**)

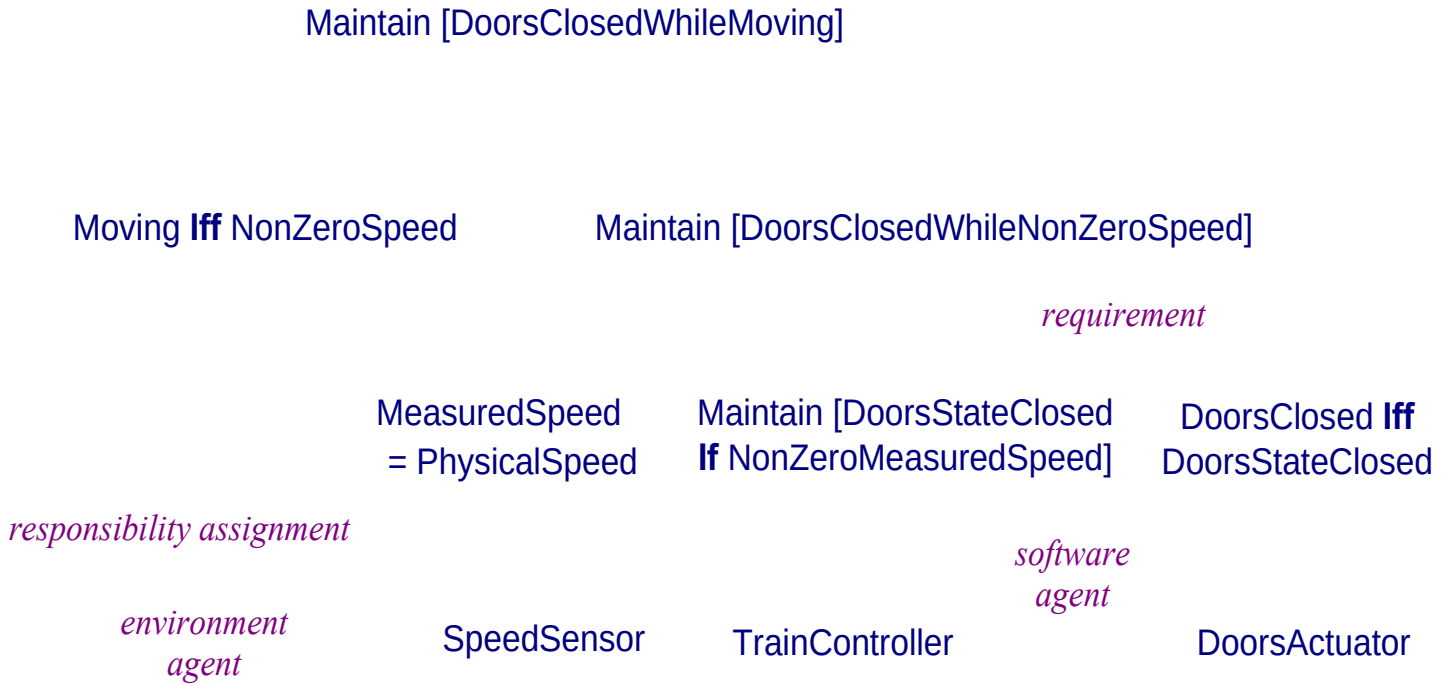
- **Minimal**: if one subgoal G_j is missing, the parent goal is no longer necessarily satisfied:

$$\{G_1, \ldots, G_{j-1}, G_{j+1}, \ldots, G_n, \text{Dom}\} \not\models G$$

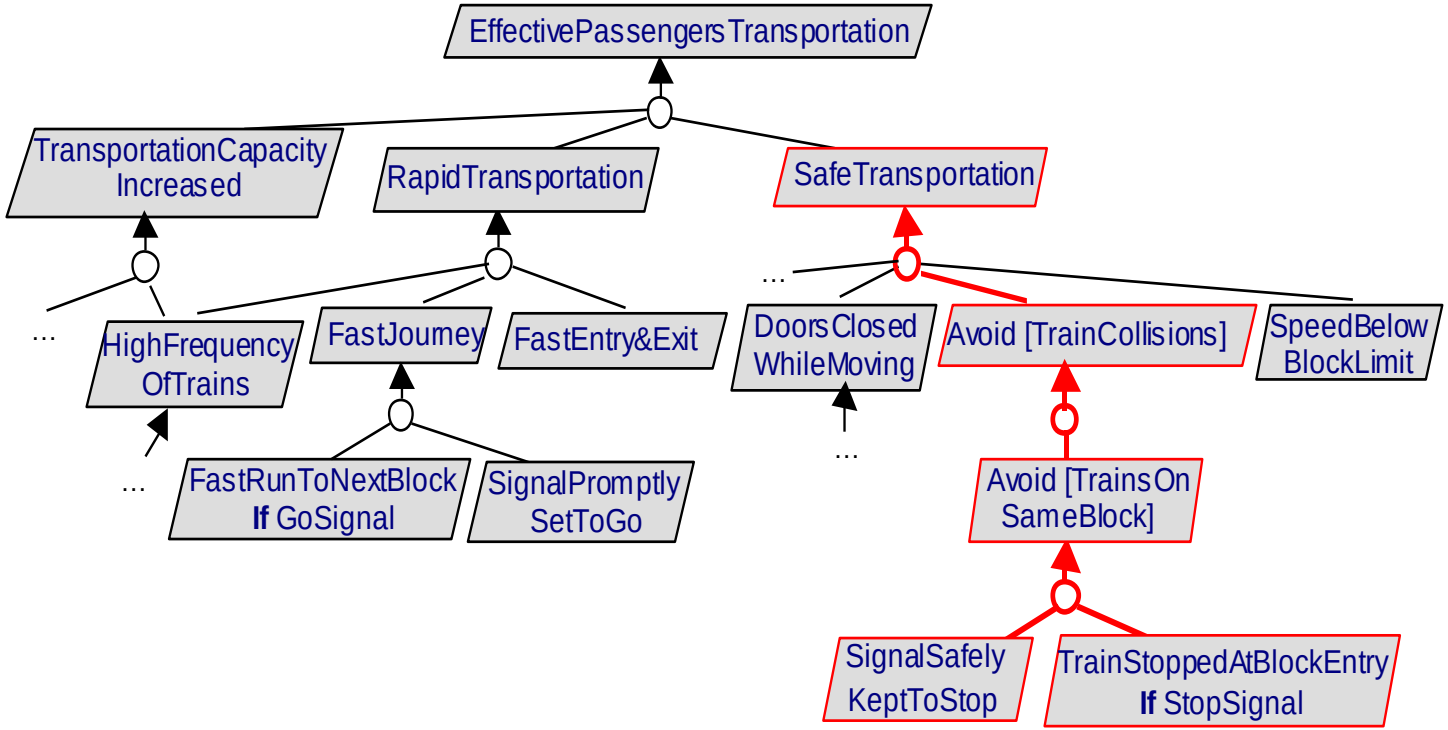
(to avoid unnecessarily restrictive requirements or expectations)

Refinement trees

- Goals are recursively refinable
- Leaf nodes = goals assignable to single system agents

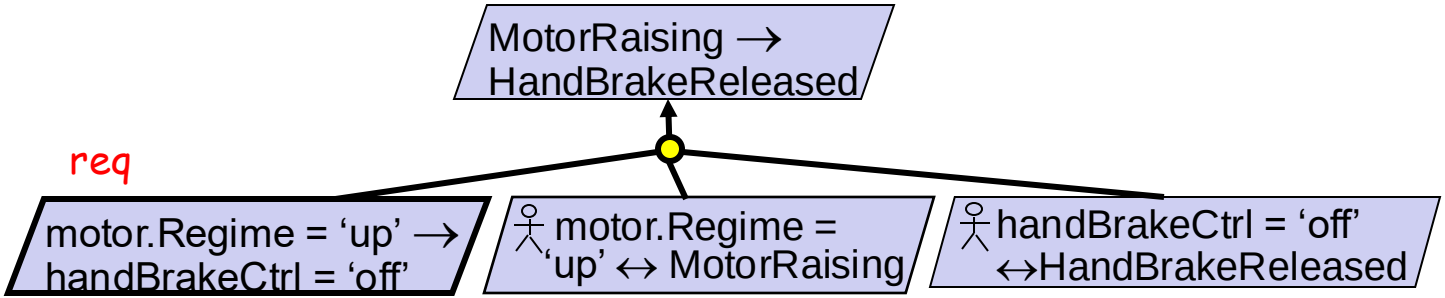


Refinement trees visualize satisfaction arguments



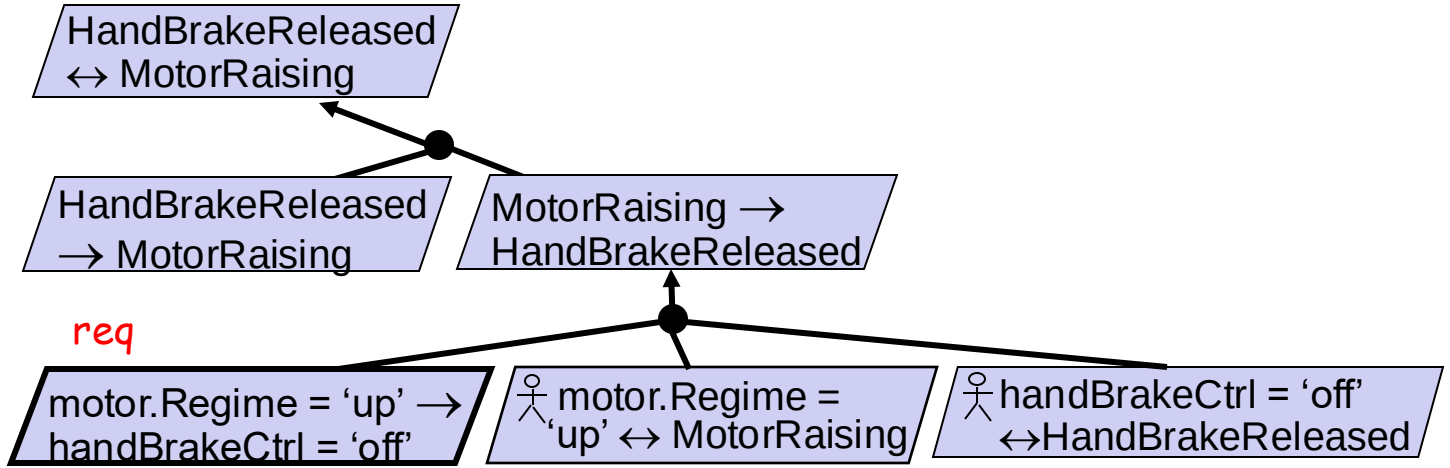
Chaining satisfaction arguments into argumentation trees

- To show how requirements ensure higher-level concerns, and recursively



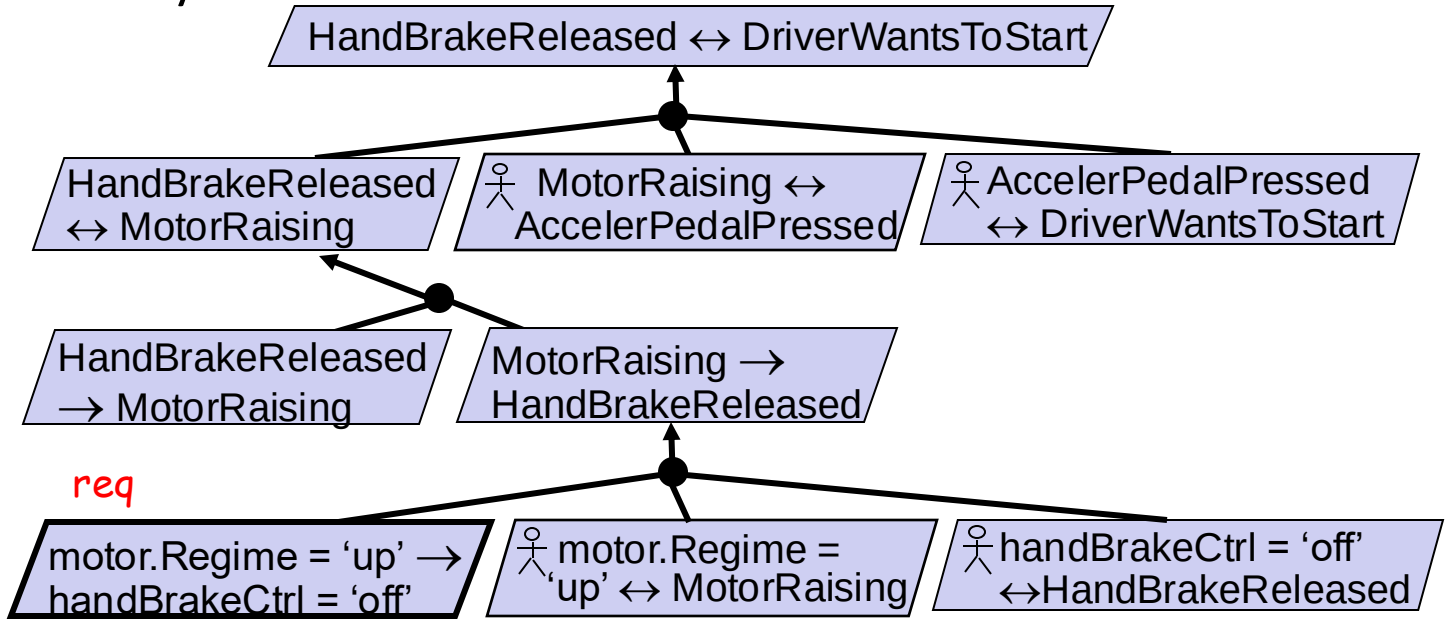
Chaining satisfaction arguments into argumentation trees

- To show how requirements ensure higher-level concerns, and recursively



Chaining satisfaction arguments into argumentation trees

- To show how requirements ensure higher-level concerns, and recursively

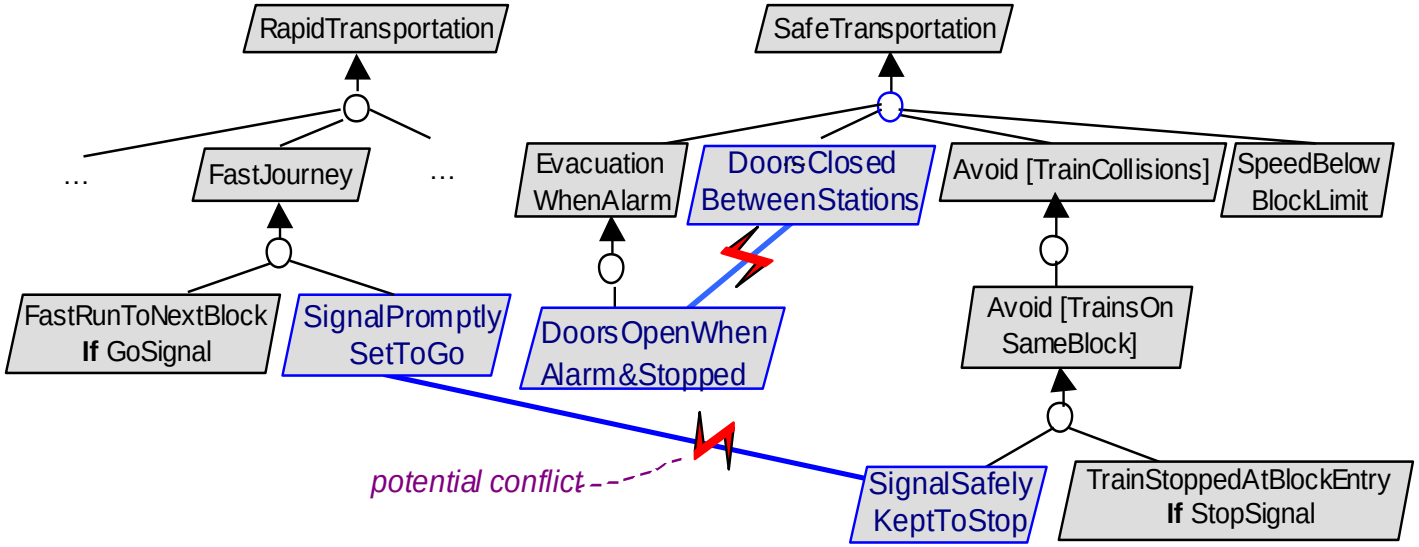


Capturing potential conflicts among goals

- Goals G_1, ..., G_n are **divergent** in *Dom* if boundary condition *B* can be found making them unsatisfiable together:

$$\{\boldsymbol{B}, G_1, \ldots, G_n, \text{Dom}\} \models \text{false}$$

- Can be captured for later analysis



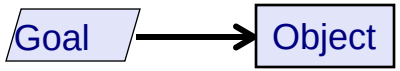
Connect the goal model with other system views

Interface links relate goals to other sub-models $\Rightarrow$ **traceability**

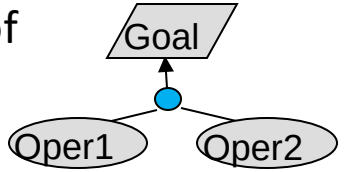
- **Responsibility**: instances of Agent are the only ones to restrict behaviors to satisfy Goal

- **Obstruction**: satisfaction of Obstacle inhibits satisfaction of Goal

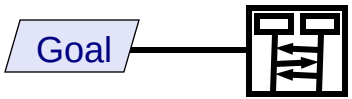
- **Concern**: specification of Goal refers to Object



- **Operationalization**: spec of Operations ensures satisfaction of Goal



- **Coverage**: behaviors prescribed by Goal cover Scenario

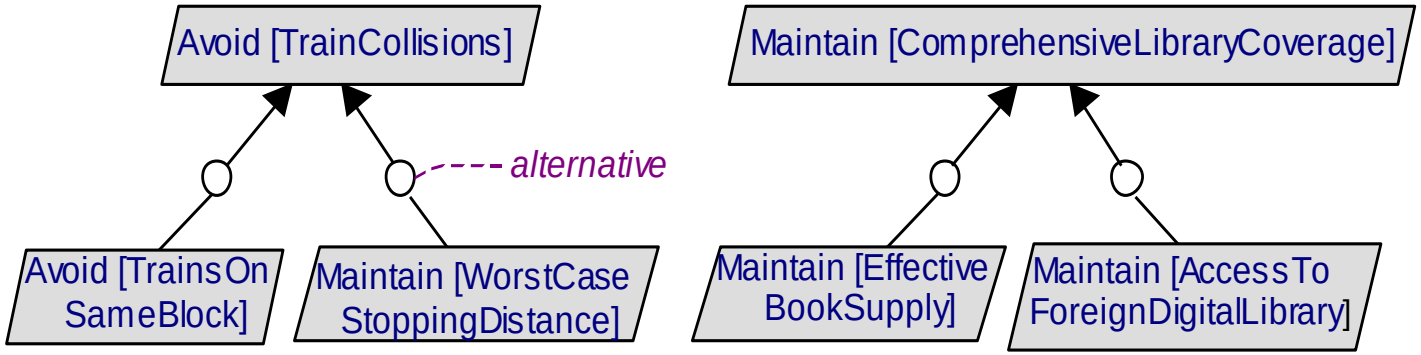


Goal modeling: outline

- Goal features as model annotations
- Goal refinement
- Capturing conflicts among goals
- Connecting the goal model with other system views
- Capturing alternative options
- Goal diagrams as AND/OR graphs
- Documenting goal refinements & assignments with annotations
- Building goal models: heuristic rules & reusable patterns

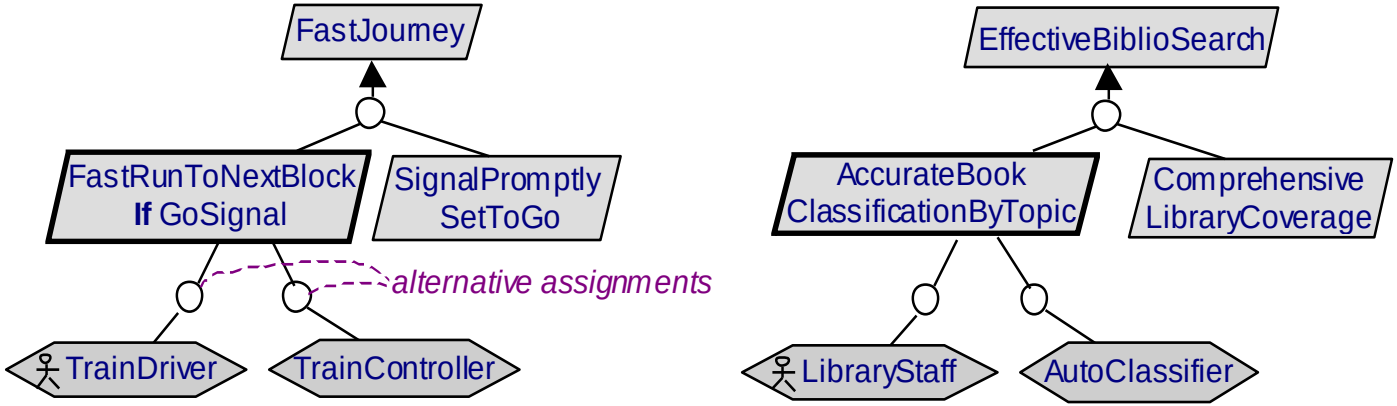
Capturing options: alternative refinements

- An **OR-refinement** of goal *G* into refinements $R_1, ..., R_m$ states that *G* can be satisfied by satisfying all subgoals from any of the alternative refinements R_i
- Alternative goal refinements yield different system proposals (variants)
- Pros/cons to be evaluated against soft goals for selection of best option



Capturing options: alternative assignments

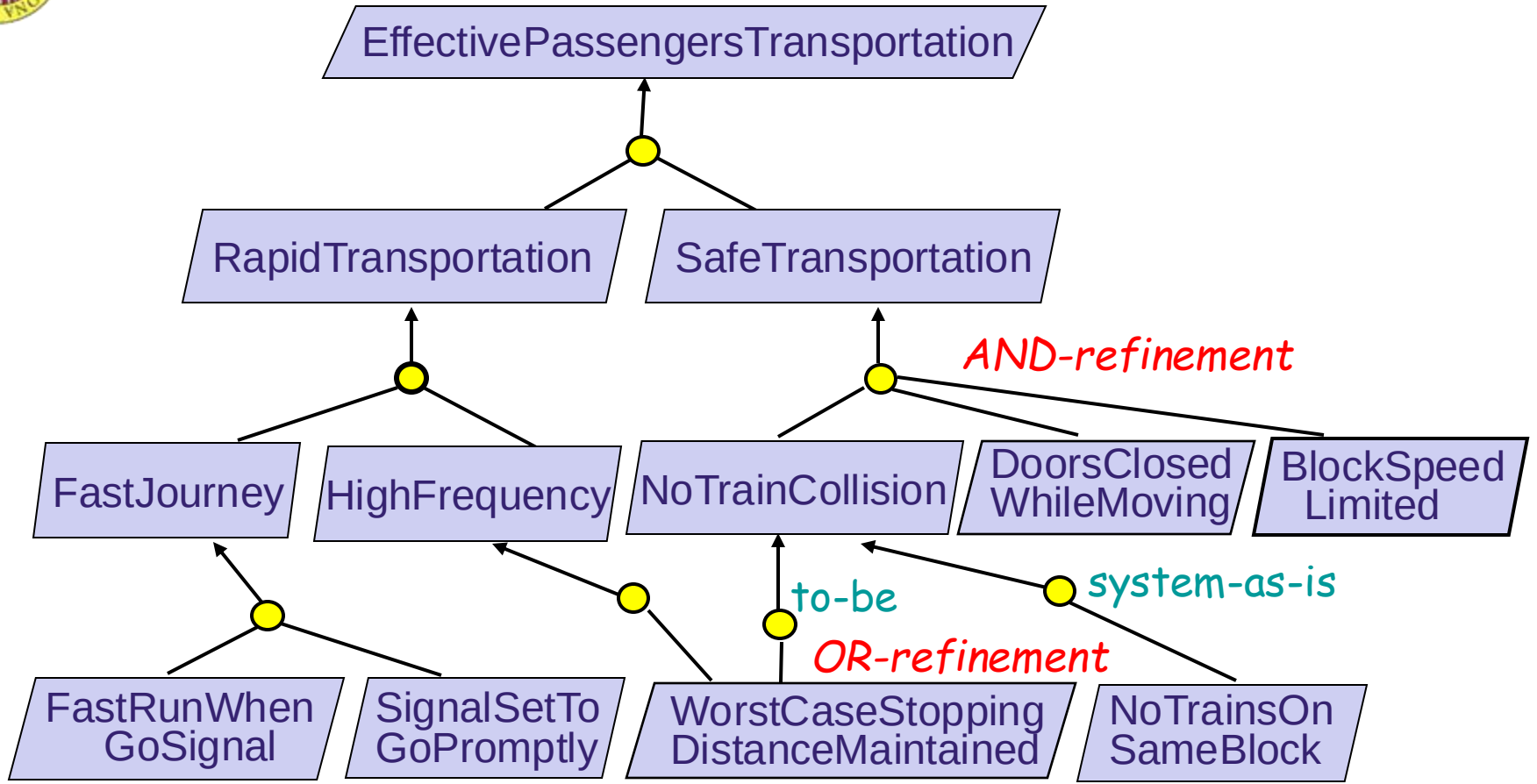
- An **OR-assignment** of goal *G* to agents A_1, ..., A_m states that *G* can be satisfied by behavioral restrictions of <u>any of the alternative</u> agents A_i
- Alternative assignments yield different system proposals
  - e.g. different degrees of automation
- Pros/cons to be evaluated against soft goals for selection of best option



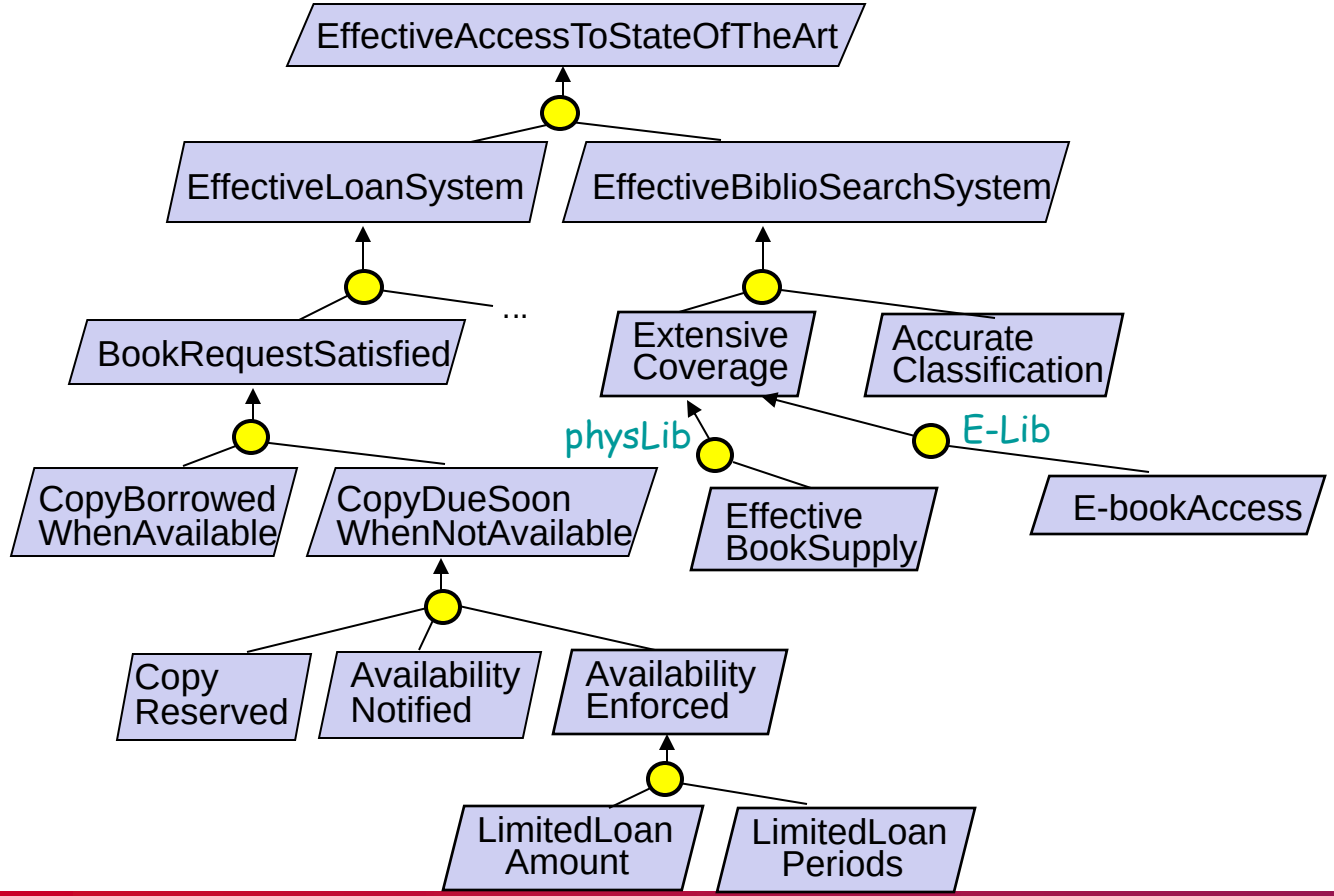
Goal diagrams as AND/OR graphs

- AND/OR graph shows how goal nodes contribute to each other
  - **roots** = high-level system goals
    - functional or non-functional
    - behavioral or soft
  - **leaves** = requirements or expectations
    - assignable to single agents
  - an **AND-refinement** links a parent goal to set of conjoined subgoals
  - an **OR-refinement** links a parent goal to a set of alternative AND-refinements => alternative system options
    - soft goals in the graph are used to select preferred options
- Generally a directed acyclic graph, not a tree
  - multiple roots (e.g. functional, non-functional goals)
  - a goal may contribute to multiple parent goals

Goal diagrams as AND/OR graphs

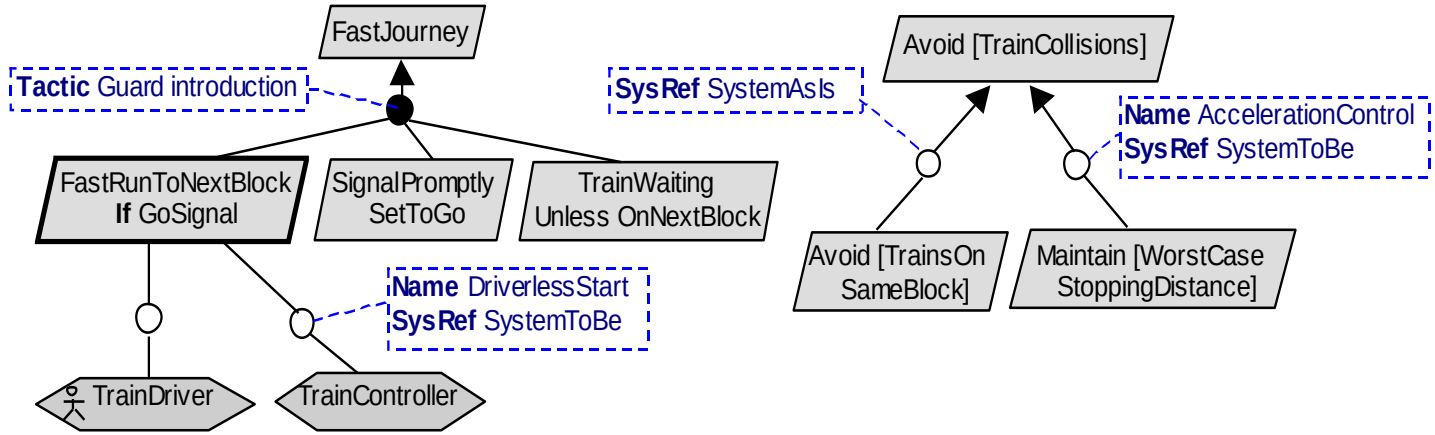


Goal diagrams as AND/OR graphs



Annotating goal refinements & assignments

- Optional features
  - **Name**: for unambiguous reference
  - **SysRef**: for associating alternatives to system versions
  - **Tactic**: for documenting refinement tactic, how it was found (cf. ref. patterns)



Goal modeling: outline

- Goal features as model annotations
- Goal refinement
- Capturing conflicts among goals
- Connecting the goal model with other system views
- Capturing alternative options
- Goal diagrams as AND/OR graphs
- Documenting goal refinements & assignments with annotations
- Building goal models: heuristic rules & reusable patterns

1. Eliciting preliminary goals

- H1: Analyze the current objectives and problems in the system-as-is
- H2: Search for goal-related keyword in elicitation material
- H3: Instantiate goal categories

Heuristic rules for early discovery of goals

- Analyze current objectives & problems in system-*as-is* ...
  - preserve strategic, organization-specific objectives & policies

    ⇒ high-level goals for system-to-be

    e.g. Effective access to state-of-the-art knowledge
  - preserve application-specific objectives to be found in any system version

    e.g. Accurate book classification
  - analyze problems & deficiencies in system-*as-is*

    ⇒ goals of system-*to-be*: Avoid / Reduce / Improve them

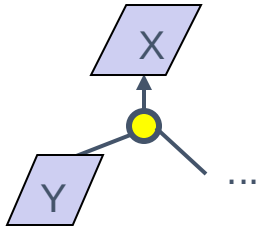
    e.g. Anywhere anytime biblio search

Heuristic rules for early discovery of goals

- Search for goal-related keywords in elicitation material (documents available, interview transcripts, etc.)
  - **intentional**: *in order to, so as to, so that, purpose, objective, aim, achieve, maintain, avoid, ensure, guarantee, want, motivate, expect,…*
  - **prescriptive**: *shall, should, must, has to, to be, may not, may never,…*
  - **amelioration**: *improve, increase, decrease, reduce, enhance, enable, support, provide, …*

  \+ *refinement links:* "***in order to*** *X* the system ***has to*** *Y* ", …

(to be checked against false positives)



Heuristic rules for early discovery of goals

- Instantiate goal categories
  - Browse leaves of taxonomies of functional & non-functional goals, looking for system-specific instances

e.g. Any **Information** goal concerning train passengers?
Any **Accuracy** goal about train information?
Any **Confidentiality** goal about meeting participants?

2. Identifying goals along refinement branches

- H4: Ask HOW and WHY questions
- H5: Split responsibilities
- H6: Identify soft goals by analyzing the pros and cons of alternative refinements
- H7: Identifying agent wishes
- H8: Analyze obstacles, threats and conflicts
- H9: Check the converse of Achieve goals
- H10: Check the complementary case of conditional Achieve goals
- H11: Refine goals until they are assignable to single agents
- H12: Abstract goals until the system's boudary is reached

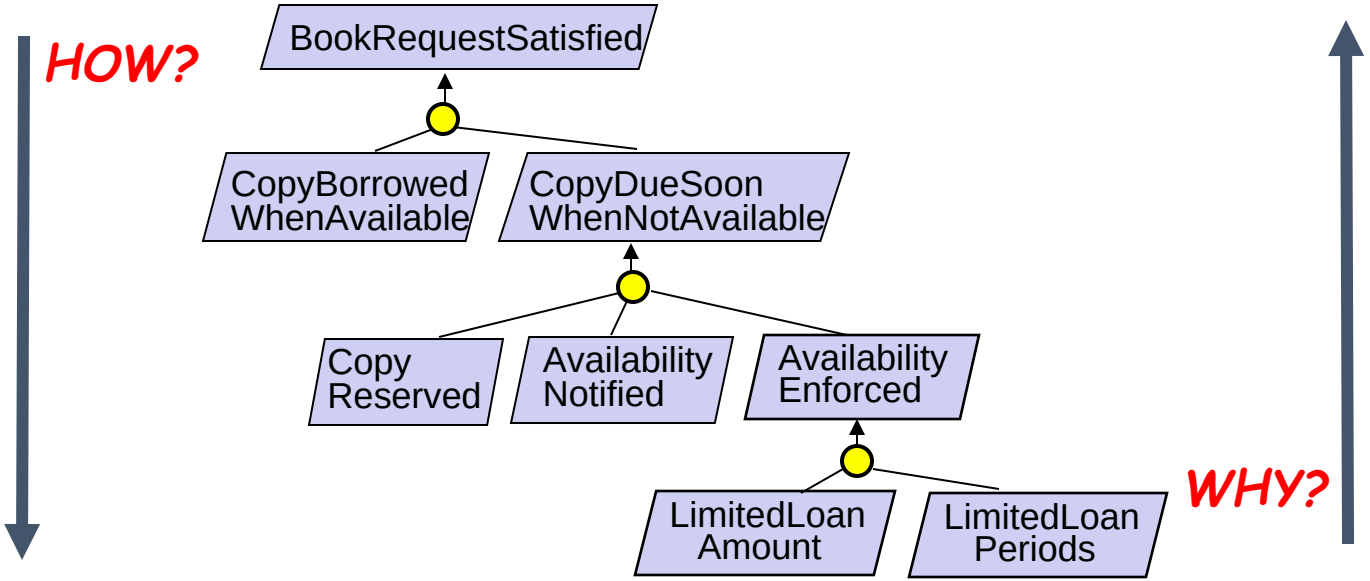
Heuristic rules for later discovery of goals

- By **abstraction** (bottom-up): ask **WHY?** questions about...
  - lower-level goals
  - interaction scenarios being elicited
  - other operational material available

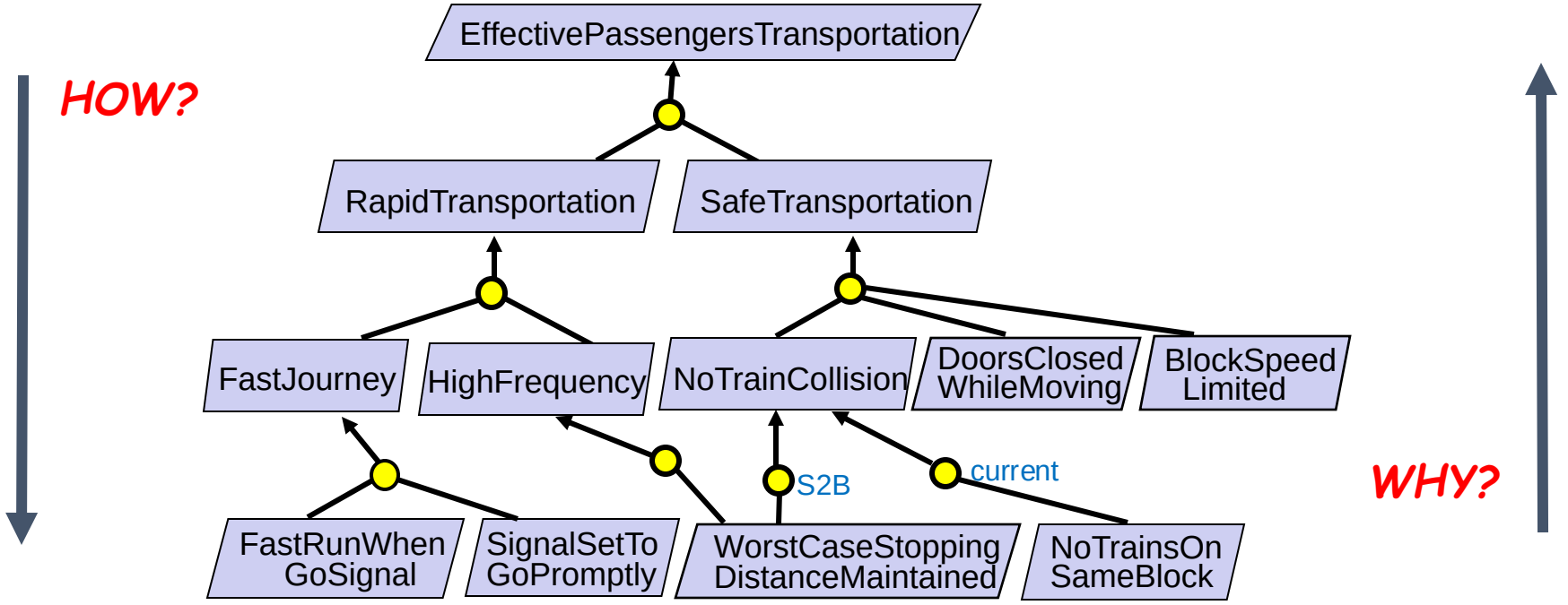
    => parent goals
- By **refinement** (top-down): ask **HOW?** questions about ...
  - higher-level-goals

    => subgoals
- Frequent questioning patterns
  - **WHY?** directly followed by **HOW?** on parent goal, to elicit missing "sibling"
  - **HOW ELSE?** to explore alternatives

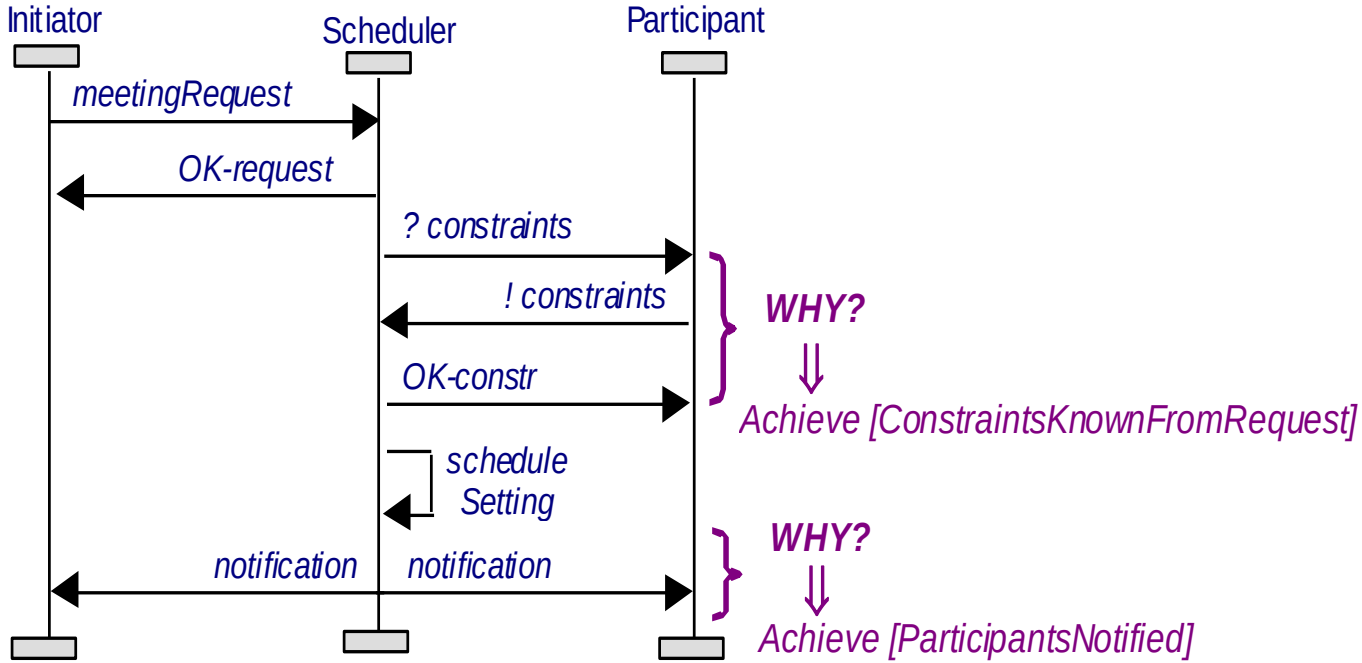
Building goal models: HOW and WHY questions



Building goal models: HOW and WHY questions

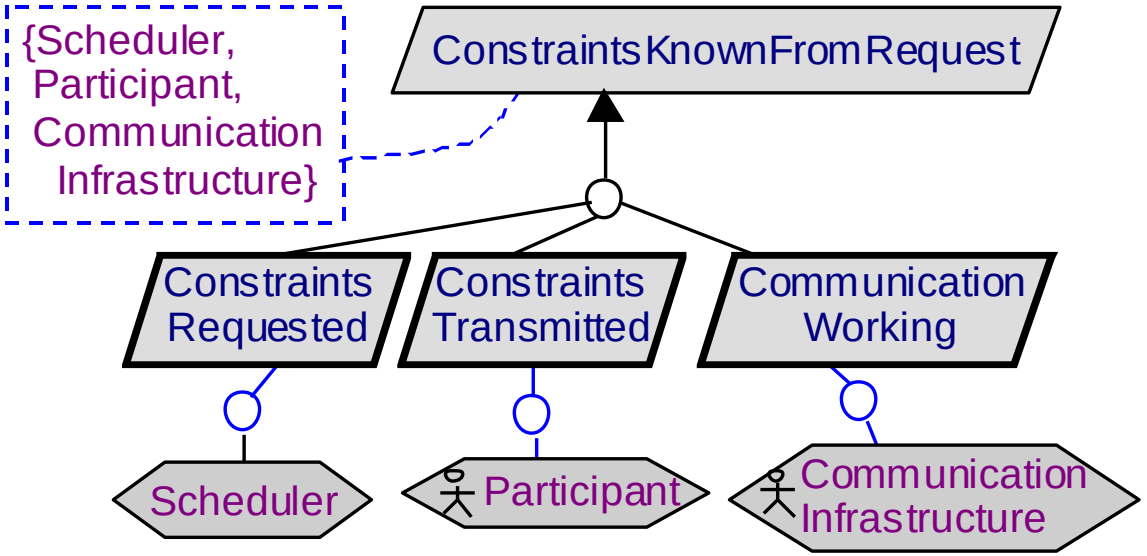


Identifying goals from WHY questions about scenario episodes



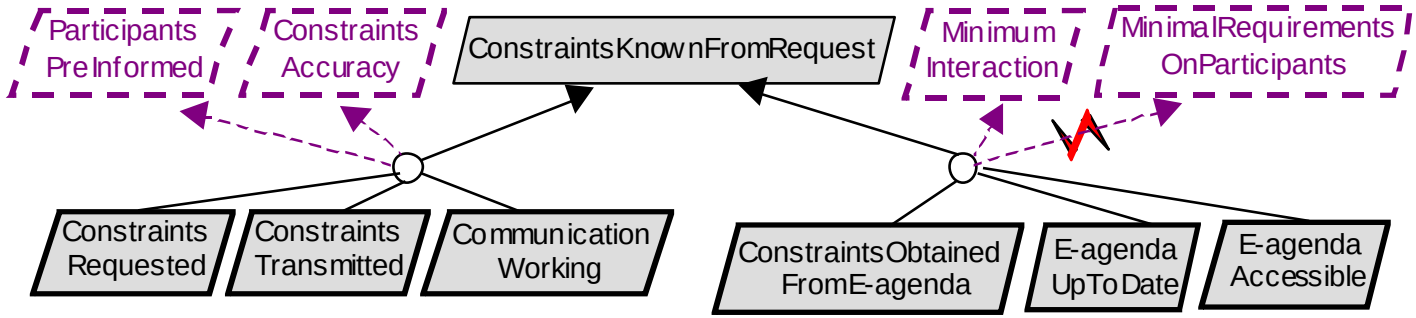
Split responsibilities among agents

To get subgoals involving fewer agents and move towards requirements and expectations



Identify soft goals from pros & cons of alternative options

- pro => **refinement** link to missing parent soft goal ?
- con => **conflict** link to missing parent soft goal ?



Heuristic rules for later discovery of goals

- Identify wishes of human agents
  - e.g. MinimalRequirementsOnParticipants

- Check the converse of Achieve goal for missing Maintain goal

  Achieve [Target **If** Condition]:
  if Condition **then sooner-or-later** Target

  ? ↓ ?

  Maintain [Target **OnlyIf** Condition]:
  always (**if** Target **then** Condition)

  e.g. Achieve [ItemSent **If** Paid] → Maintain [ItemSent **OnlyIf** Paid]
  Achieve [reverseThrustEnabled **If** PlaneOnGround]
  → Maintain [reverseThrust **OnlyIf** PlaneOnGround]

Building goal models: delimiting their scope

- **Refine** goals … *until when ?*

  … until assignable to **single** agents as …

  - **requirement** (software agent)
  - **expectation** (environment agent)

- **Abstract** goals … *until when ?*

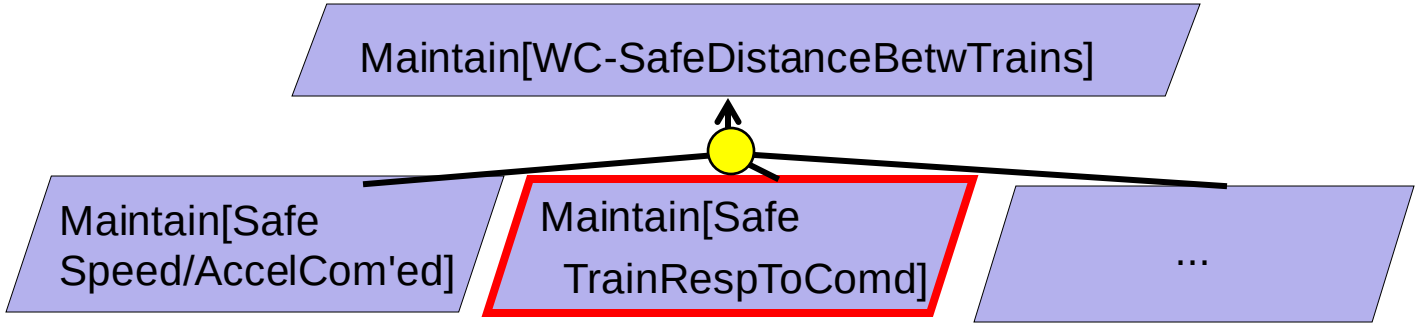
  … until boundary of system capabilities is reached:

  goals that cannot be satisfied solely by system agents

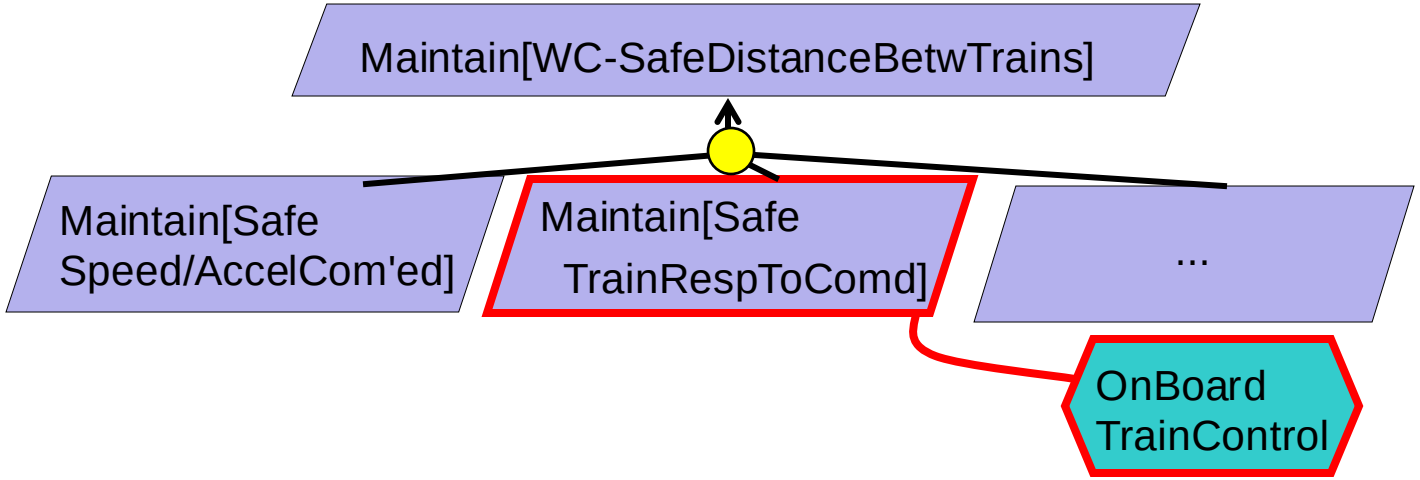
  e.g. EliminateGreenhouseEffect

  is beyond capabilities of train system

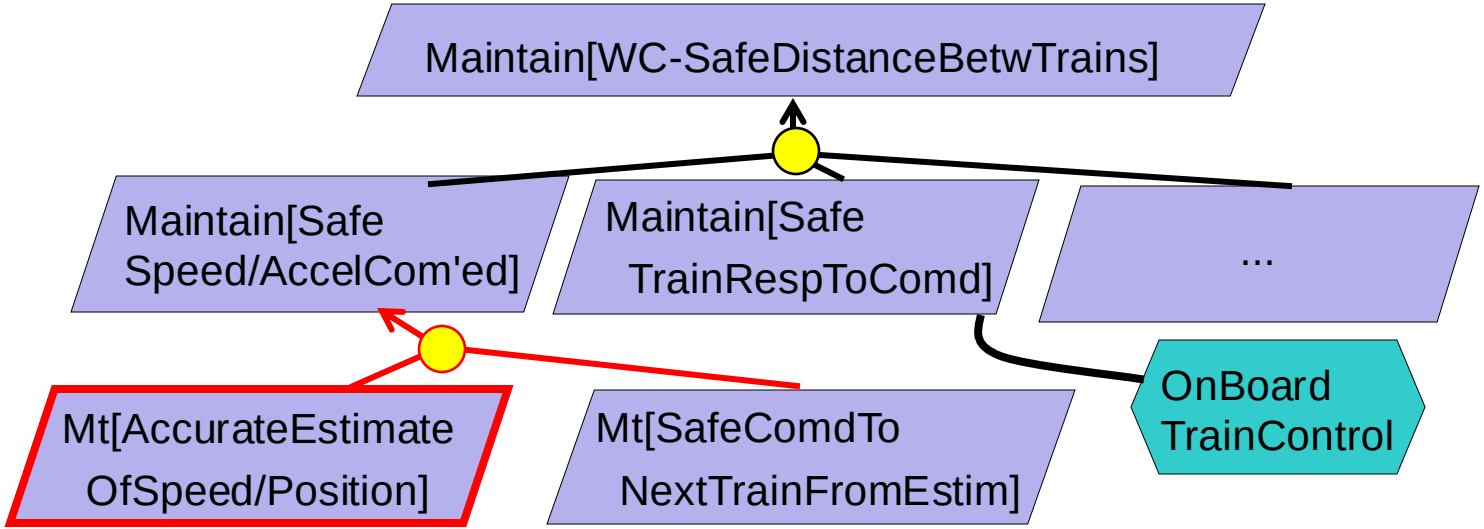
Goal refinement … until when ?



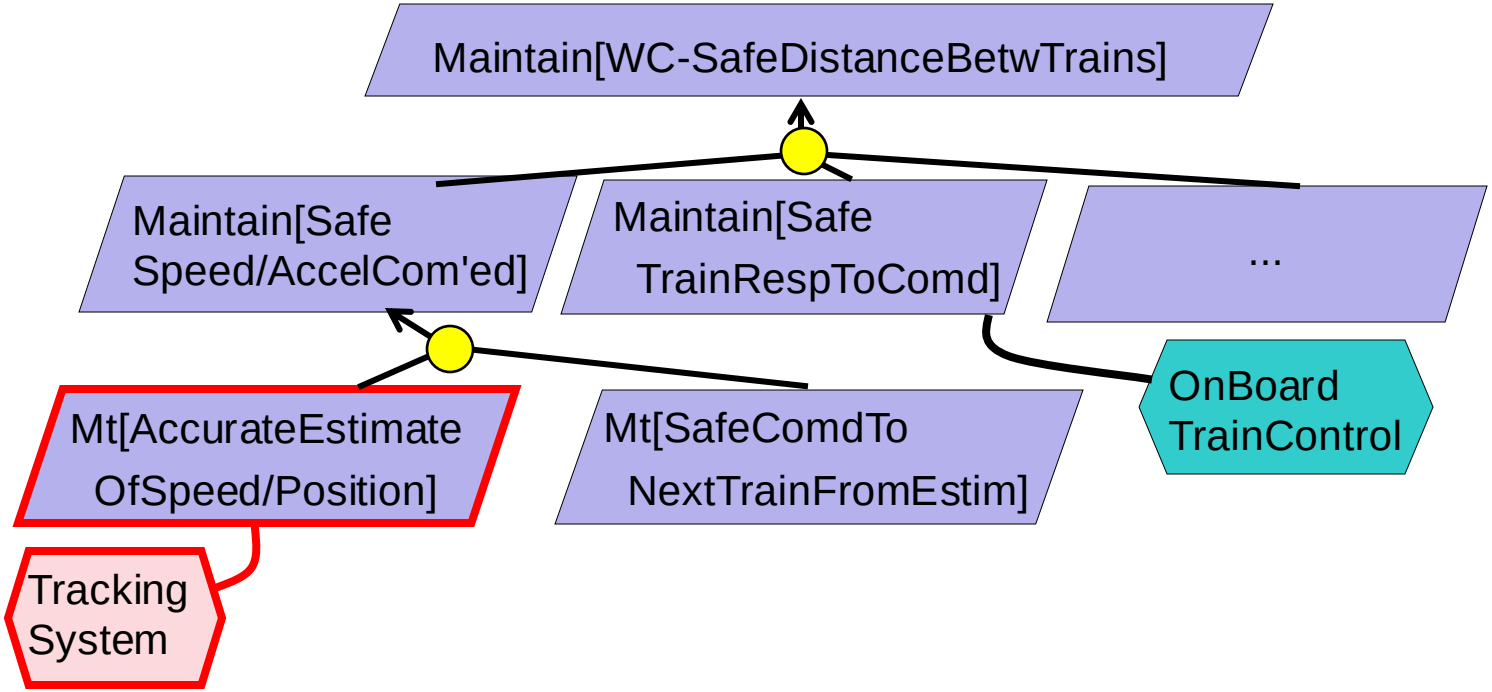
Goal refinement … until when ?



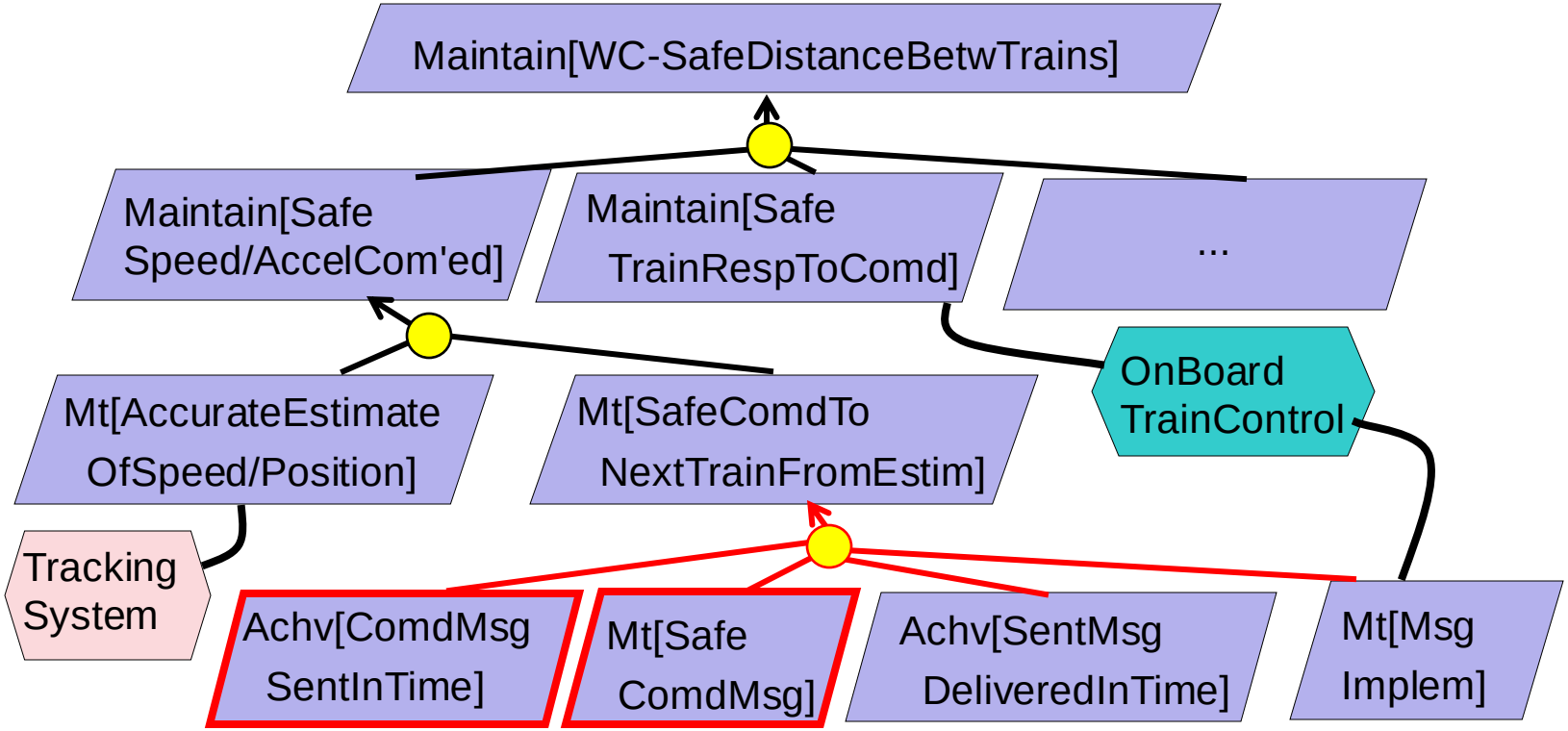
Goal refinement … until when ?



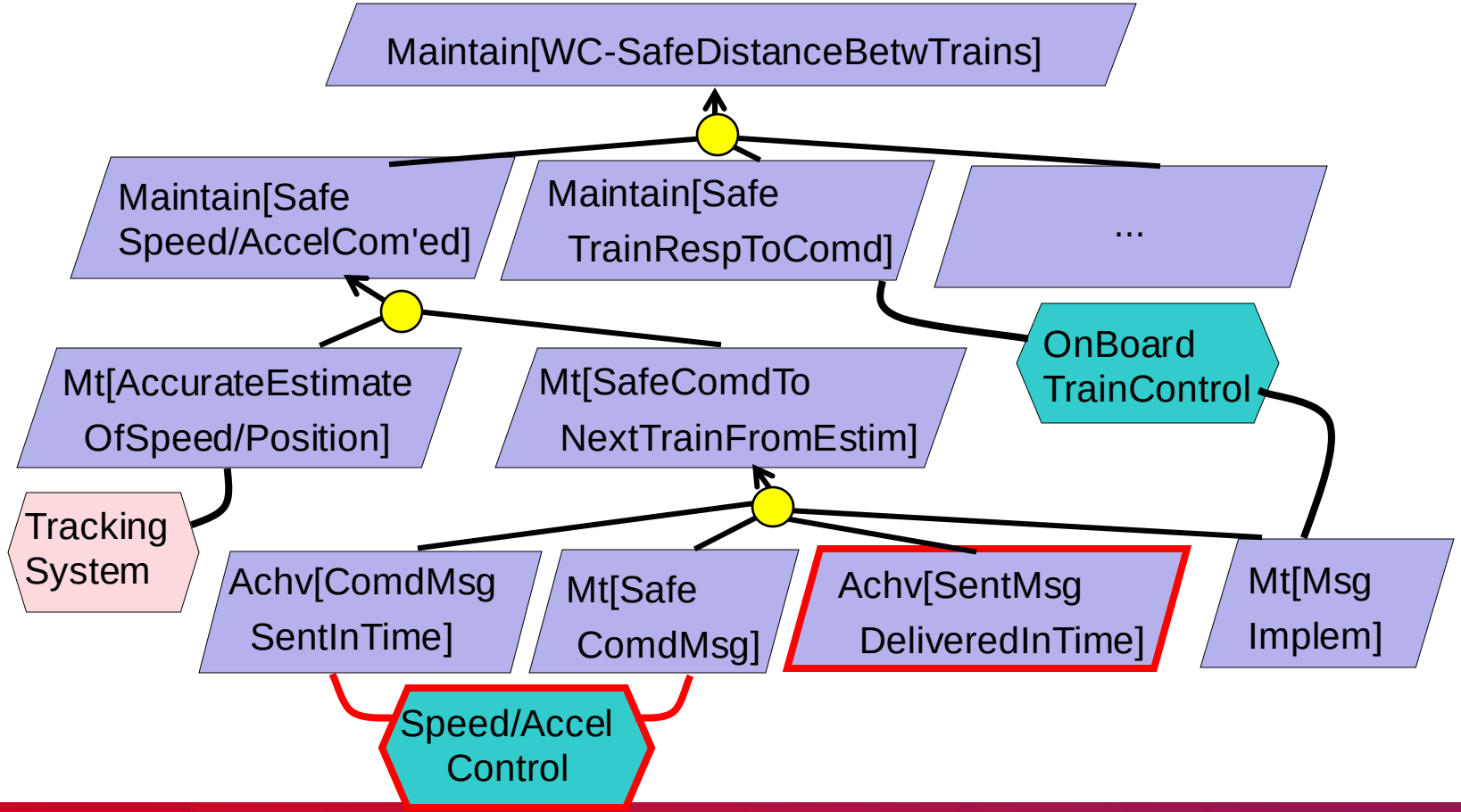
Goal refinement … until when ?



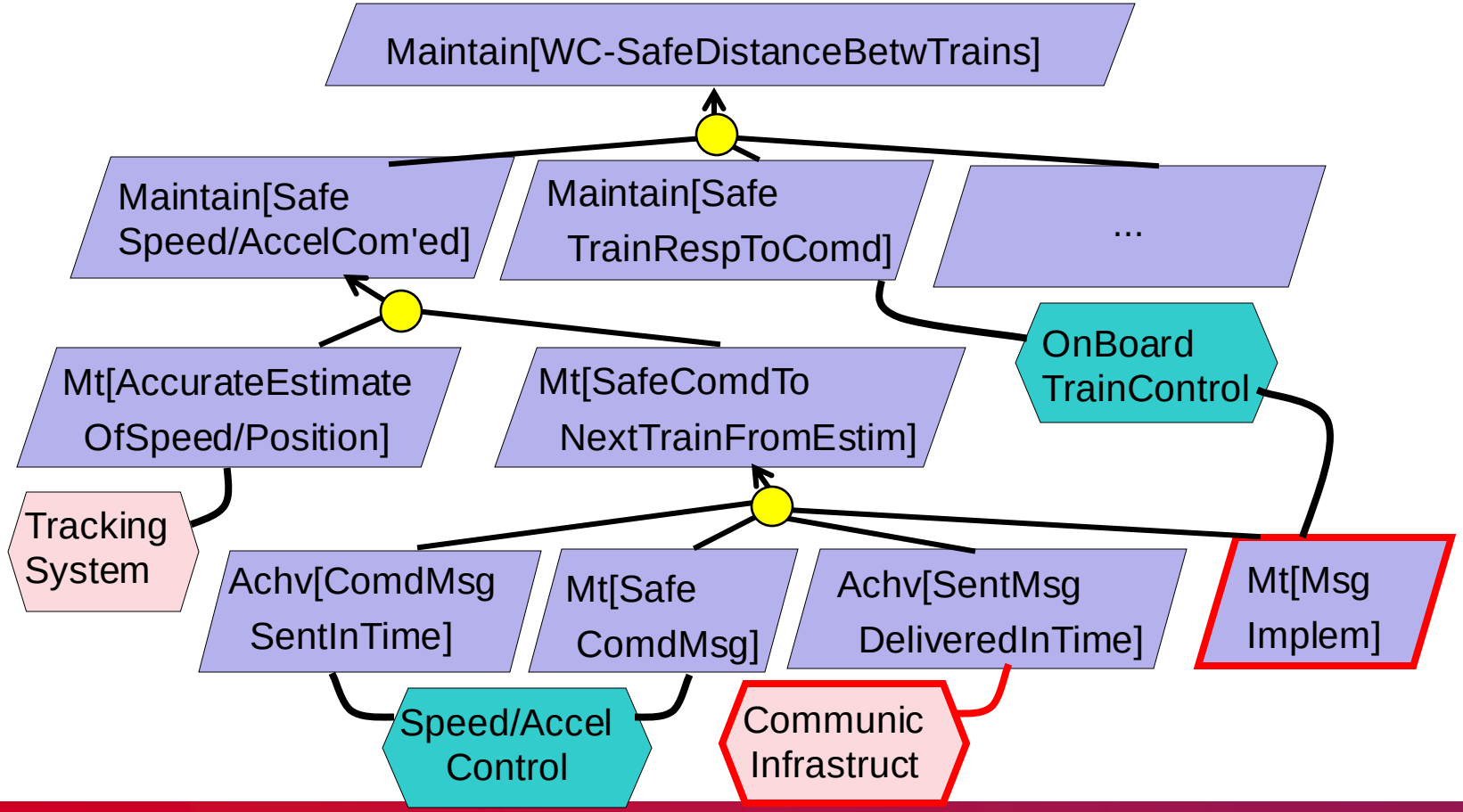
Goal refinement … until when ?



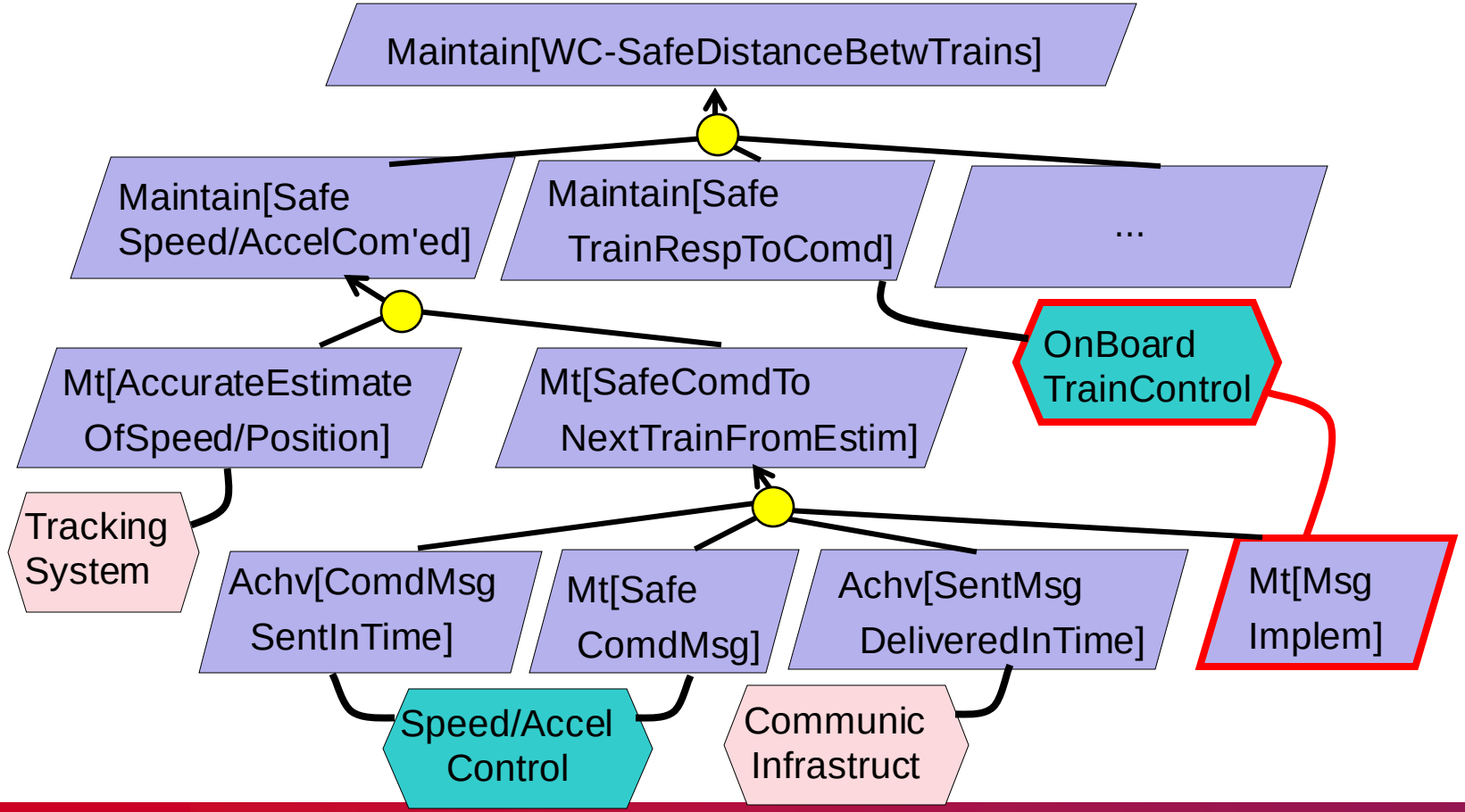
Goal refinement … until when ?



Goal refinement … until when ?



Goal refinement … until when ?



3. Avoid common pitfalls: Bad smells

- H13: Do not confuse goals with operations
- H14: Do not confuse AND-refinements (cases) with OR-refinements (alternatives)
- H15: Avoid ambiguities in goal specifications

Do not confuse goals with operations

- Do not confuse ...

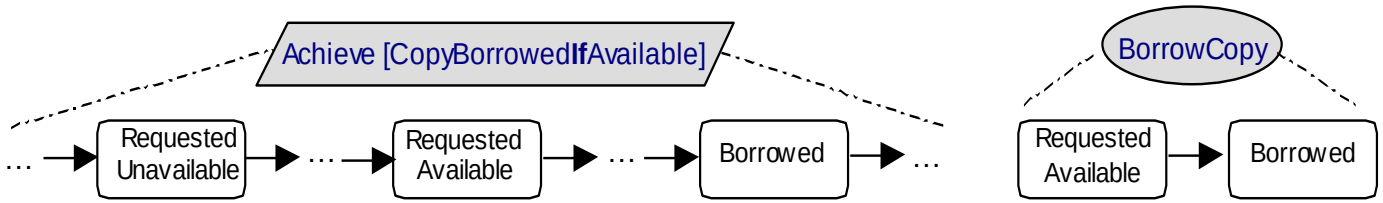
goal ...

operation ...

- Goal ≠ service from functional model (e.g. use case)
- Services **operationalize** functional, leaf goals in refinement graph
  - a goal is often operationalized through multiple operations
  - an operation often operationalizes multiple goals
- Soft goals are often *not* operationalized in functional model but used to select among alternatives

Behavioral goals vs. operations

- Semantic difference
  - Behavioral goals constrain entire sequences of state transitions
  - Operations constrain single state transitions

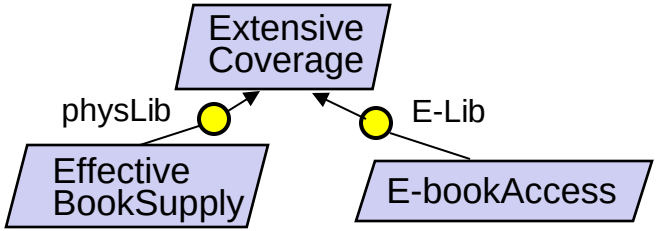


- **Tip**: use <u>past participle</u> for goal name
  (state to be reached/maintained, quantity to be reduced/increased, ...)
  use <u>infinitive</u> for operation name
  (action to reach/maintain that state)

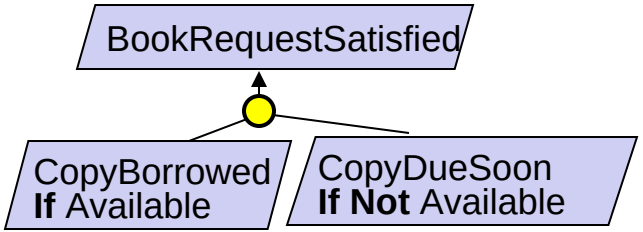
Do no confuse AND-/OR-refinements

- Do not confuse ...

OR-refinement ...



AND-refinement by case ...



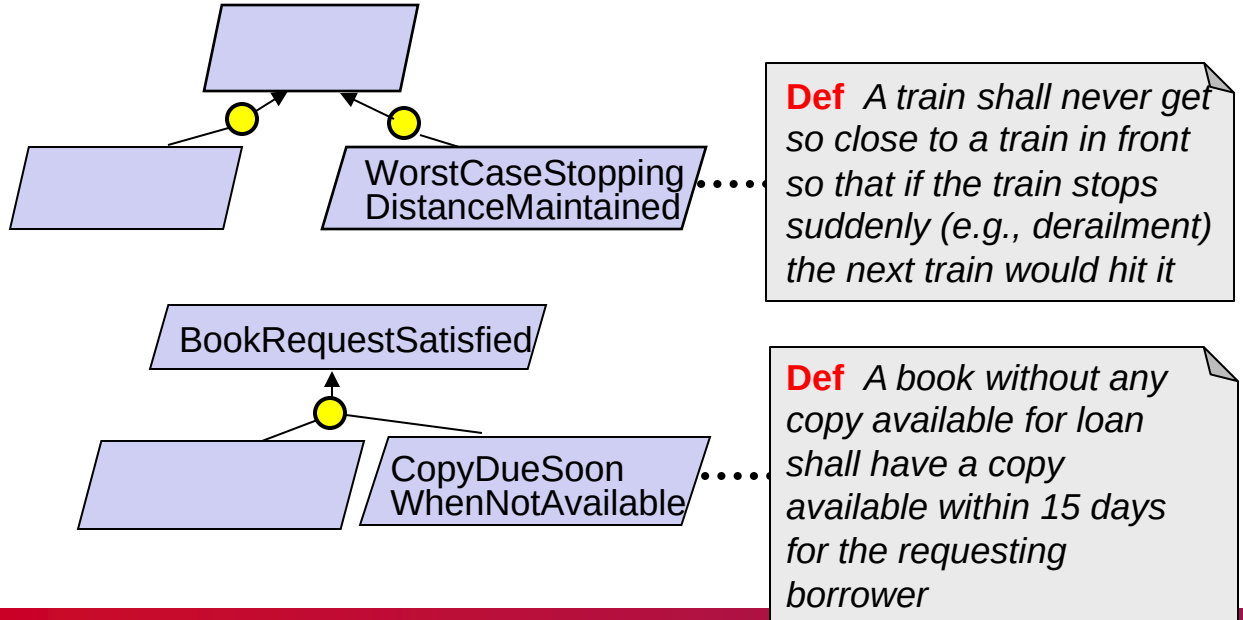
cf. case analysis:

(Case1 **or** Case2) $\Rightarrow$ X equiv (Case1 $\Rightarrow$ X) **and** (Case2 $\Rightarrow$ X)

- **OR-**refinement introduces alternative systems to reach parent goal
- **AND-**refinement by cases introduces complementary, conjoined subgoals within same system

Avoid ambiguities in goal specification

- Avoid ambiguities in goal specification & interpretation ...
  - a precise & complete goal **definition** is essential
  - grounded on shared system phenomena, and agreed upon by all stakeholders



Building goal models: Reuse refinement patterns

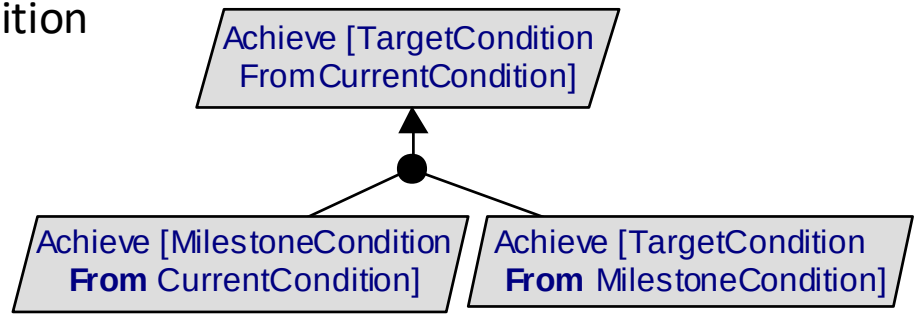
Reuse refinement patterns

From a catalogue of generic, complete AND-refinements:

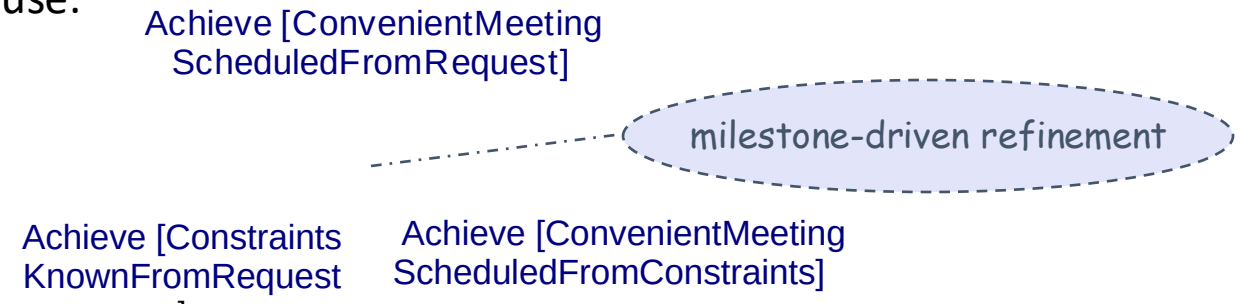
- encode refinement **tactics**, codify modeller's experience
- guide modeling process by suggesting refinements to be instantiated
  - instantiated pattern may sometimes require adaptation
- provide satisfaction argument for free
  - (formal) correctness proof done once for all, kept hidden
- support model documentation & understanding
- can also be used for:
  - completing partial refinements
  - exploring alternative options (multiple applicable patterns)

A sample of refinement patterns

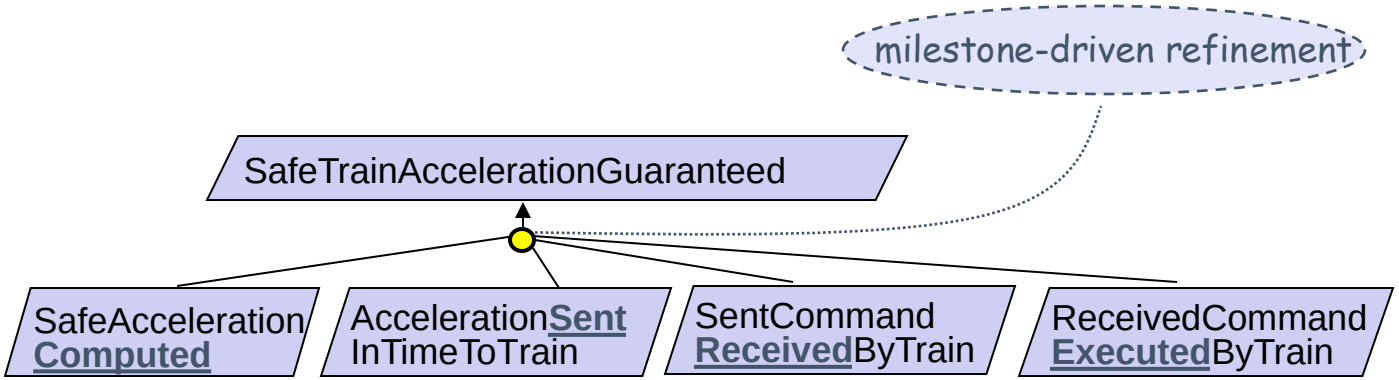
- Refinement by milestone
  - Applicable when milestone states can be identified on the way to the goal's target condition



  - Example of use:

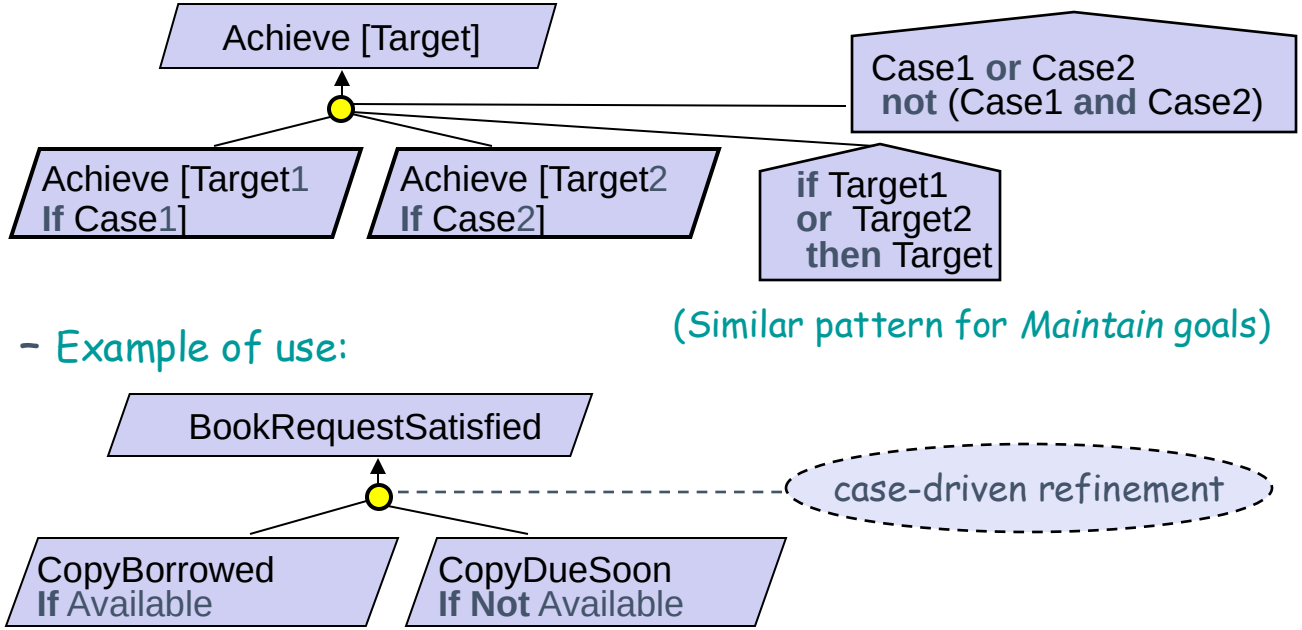


Variant for Maintain goals



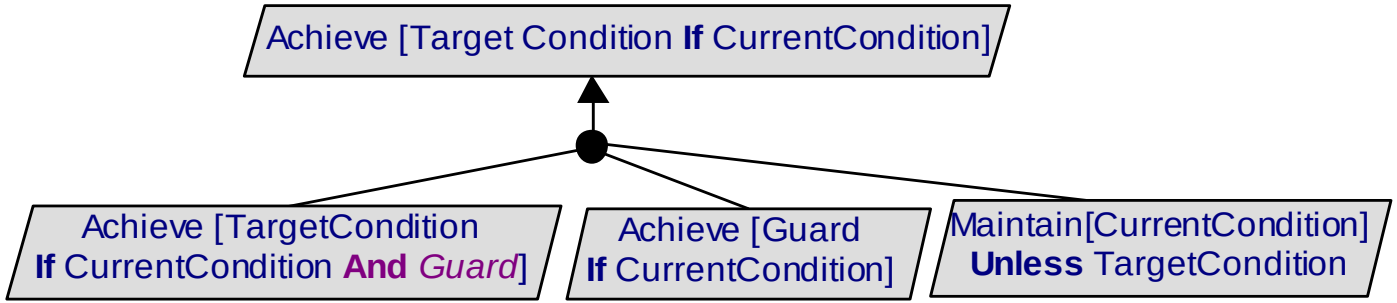
Refinement by case

- Applicable when the goal satisfaction space can be partitioned into cases (disjoint, covering all possibilities)

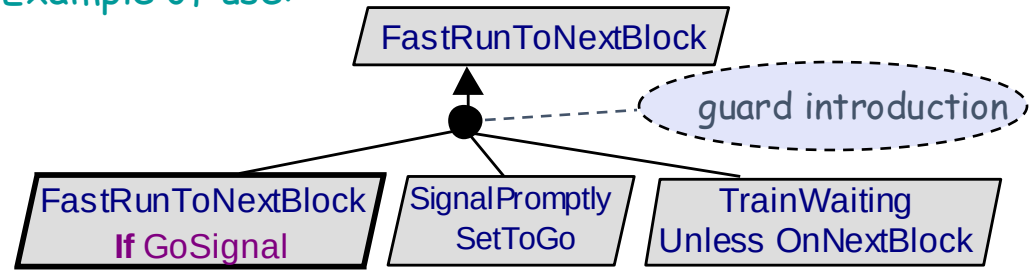


Guard introduction

- Applicable to *Achieve* goals where a guard condition must be set for reaching the target

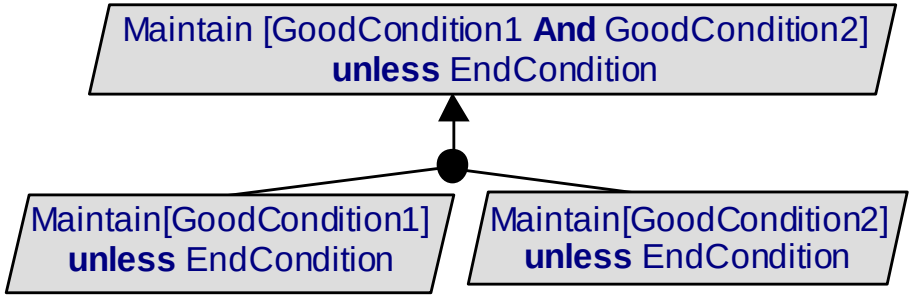


- Example of use:

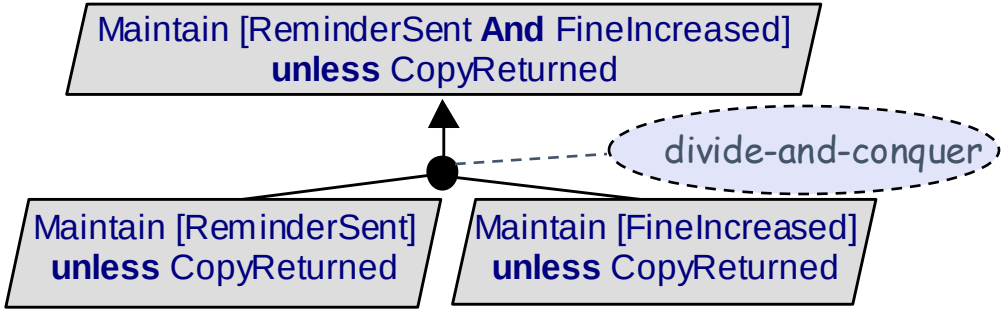


Divide-and-Conquer

- Applicable to *Maintain* goals where GoodCondition is a conjunction

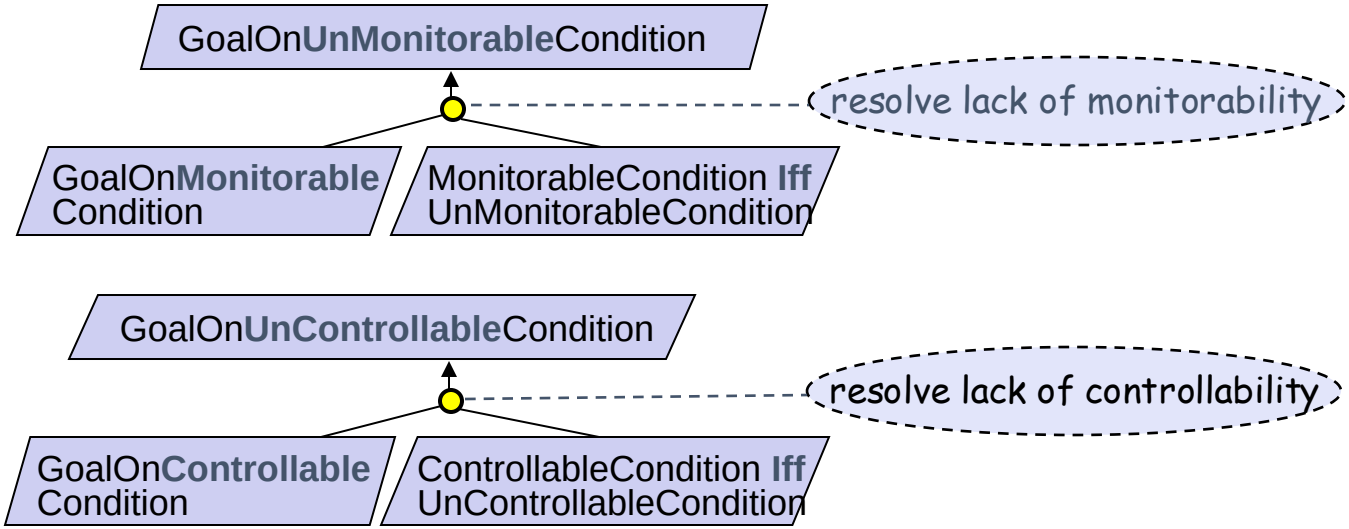


- Example of use:



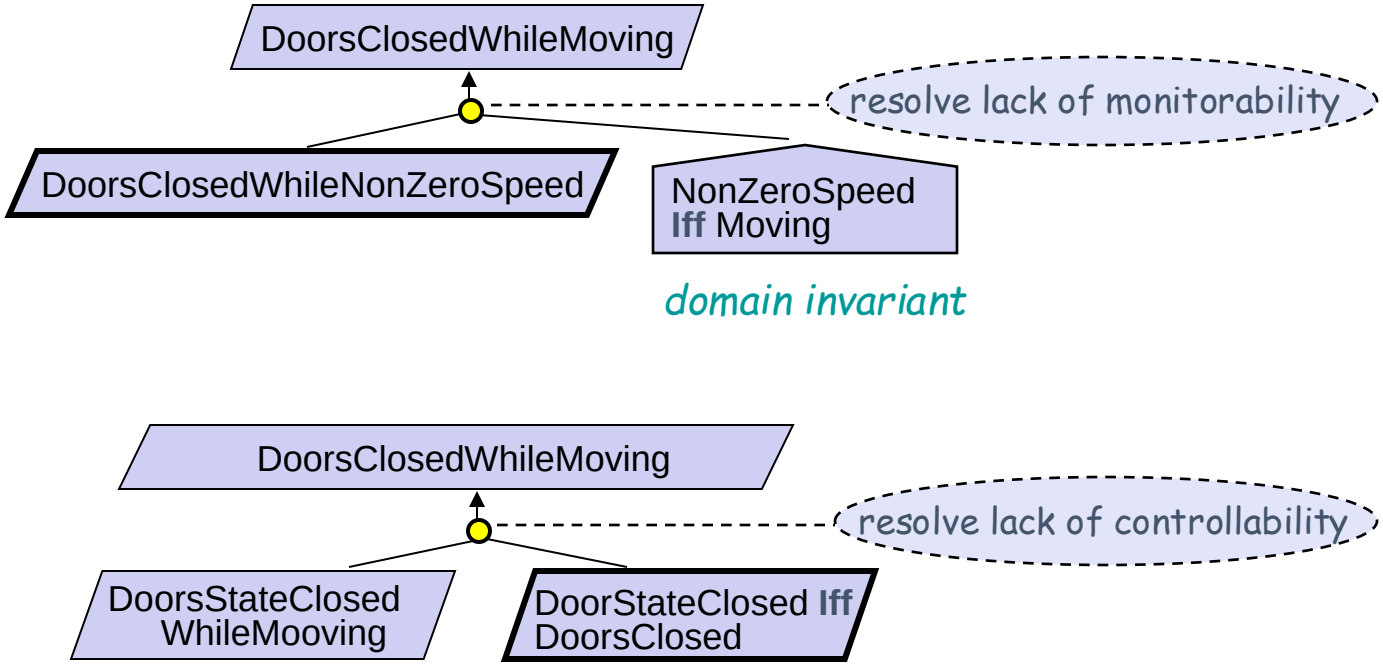
Refinement towards goal realizability

- Applicable when the goal refers to quantities not monitorable or not controllable by candidate agent

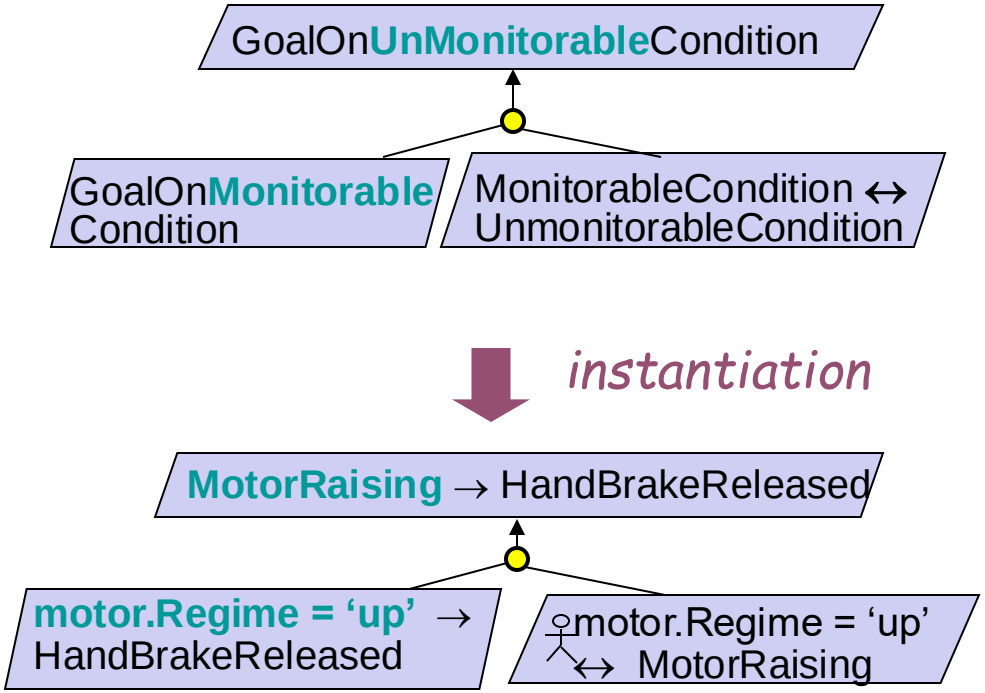


Child node may be goal (incl. requirement, expectation)
or domain property (invariant/hypothesis)

Refinement towards goal realizability: examples of use



Refinement towards goal realizability: examples of use



Refinement towards goal realizability: examples of use