Problem Frame Transformations: Deriving Specifications from Requirements

Robert Seater
Daniel Jackson

Software Design Group
Massachusetts Institute of Technology

May 23rd, 2006
2nd International Workshop on Applications and Advances in Problem Frames (IWAAPF'06, part of ICSE'06)
History

- tool for understanding proton therapy machine
- understand & analyze existing system
- handle arbitrary topologies
- local reasoning (local understanding)
- record reasoning (tracability, communication)
- experts on hand
Requirements & Specifications

• does the spec enforce the requirement?
Requirements & Specifications

• does the spec enforce the requirement?
• relies on **domain assumptions**
Requirements & Specifications

- does the spec enforce the requirement?
- relies on **domain assumptions**
- conventional solution: catalogue of **frame concerns** derived from prior experience
- template for correctness argument, list of relevant assumptions
Key Observations

- requirement is not a spec only because it references phenomena not controlled by the machine
- domain assumption justifies constraining p1 instead of p2
- can incrementally transform requirement into spec plus set of domain assumptions
Transformation Toolkit

- **add** a breadcrumb
- **rephrase** the goal
- **push** an arc
- **split/merge** arcs
- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
- **rephrase** the goal
- **push** an arc
- **split/merge** arcs
- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  
  *relate referenced phenom to other phenom*

- **rephrase** the goal

- **push** an arc

- **split/merge** arcs

- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  
  *relate referenced phenom to other phenom*

- **rephrase** the goal

- **push** an arc

- **split/merge** arcs

- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  
  relate referenced phenom to other phenom

- **rephrase** the goal

  Breadcrumb $\wedge$ Rephrased Goal $\Rightarrow$ Prior Goal

- **push** an arc

- **split/merge** arcs

- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  
  relate referenced phenom to other phenom

- **rephrase** the goal
  
  \[\text{Breadcrumb} \, ^{\wedge} \, \text{Rephrased Goal} \rightarrow \text{Prior Goal}\]

- **push** an arc

- **split/merge** arcs

- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  
  relate referenced phenom to other phenom

- **rephrase** the goal
  
  $\text{Breadcrumb} \ ^\wedge \text{Rephrased Goal} \Rightarrow \text{Prior Goal}$

- **push** an arc
  
  phenom on that arc must be shared

- **split/merge** arcs

- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  relate referenced phenom to other phenom

- **rephrase** the goal
  \[\text{Breadcrumb} \, ^\wedge \, \text{Rephrased Goal} \Rightarrow \text{Prior Goal}\]

- **push** an arc
  phenom on that arc must be shared

- **split/merge** arcs

- **heuristic**: walk the requirement towards the machine
Transformation Toolkit

- **add** a breadcrumb
  relate referenced phenom to other phenom

- **rephrase** the goal
  Breadcrumb $\wedge$ Rephrased Goal $\Rightarrow$ Prior Goal

- **push** an arc
  phenom on that arc must be shared

- **split/merge** arcs
  nothing else changes

- **heuristic**: walk the requirement towards the machine
Two-Way Traffic Light
Two-Way Traffic Light

Northward

Southward
Two-Way Traffic Light

Northward

Southward
Two-Way Traffic Light

Northward

Southward
Two-Way Traffic Light

NGpulse

NGobserve

Northward

Southward
Two-Way Traffic Light

NGpulse

SRобserve

Northward

Southward
Two-Way Traffic Light

NRpulse

NRobserve

SRobserve

Northward

Southward
Two-Way Traffic Light

NGpulse

NGobserve

NRobserve

Southward

Northward
Problem Frame Description

no t: time | some c1, c2 : Cars | CarDirection(c1, t) = north and CarDirection(c2, t) = south and CarOnSegment(c1, t) and CarOnSegment(c2, t)
**Rephrase 1**

- Control Unit
  - NRpulse
  - NGpulse
  - SRpulse
  - SGpulse

- Light Unit
  - NGobserve
  - SGobserve

- Cars
  - CarDirection
  - CarOnSegment
  - NGobserve
  - SGobserve

- No t: time | NGobserve(t) and SGobserve(t)
- No t: time | some c1, c2 : Cars | CarDirection(c1, t) = north and CarDirection(c2, t) = south and CarOnSegment(c1, t) and CarOnSegment(c2, t)

- All t: time | ! NGobserve(t) => no c: Cars | CarDirection(c, t) = north and CarOnSegment(c, t)
- All t: time | ! SGobserve(t) => no c: Cars | CarDirection(c, t) = south and CarOnSegment(c, t)
no t: time I NGobserve(t) and SGobserve(t)

all t: time I! NGobserve(t) =>
  no c: Cars I
  CarDirection(c, t) = north and CarOnSegment(c, t)

all t: time I! SGobserve(t) =>
  no c: Cars I
  CarDirection(c, t) = south and CarOnSegment(c, t)
Push 1

- Control Unit
  - NRpulse
  - NGpulse
  - SRpulse
  - SGpulse

- Light Unit
  - NGobserve
  - SGobserve

- Cars
  - CarDirection
  - CarOnSegment
  - NGobserve
  - SGobserve

- no t: time \( t \) ! NGobserve\( (t) \) and SGobserve\( (t) \)

- all t: time \( t \) ! NGobserve\( (t) \) => no c: Cars \( c \) CarDirection\( (c, t) = \text{north} \) and CarOnSegment\( (c, t) \)

- all t: time \( t \) ! SGobserve\( (t) \) => no c: Cars \( c \) CarDirection\( (c, t) = \text{south} \) and CarOnSegment\( (c, t) \)
Breadcrumb 2
Push 2

**System Description**

- **Control Unit**
  - Input: NRpulse, NGpulse, SRpulse, SGpulse

- **Light Unit**
  - Input: NRobserve, NGobserve, SGpulse
  - Output: C

- **Cars**
  - Input: CarDirection, CarOnSegment, NGobserve, SGobserve

**Rules**

- All t: time 1
  - NGobserve(t) ↔ odd(NGpulse, t) and SGobserve(t) ↔ odd(SGpulse, t)

- No t: time 1
  - odd(NGpulse, t) and odd(SGpulse, t)

- All t: time 1 ! NGobserve(t) ⇒ no c: Cars 1 CarDirection(c, t) = north and CarOnSegment(c, t)

- All t: time 1 ! SGobserve(t) ⇒ no c: Cars 1 CarDirection(c, t) = south and CarOnSegment(c, t)
Provides

- systematic local reasoning
  \[ Breadcrumb ^ \text{Rephrased Goal} \Rightarrow \text{Prior Goal} \]
- global guarantee
  \[ Breadcrumb_{0} ^ \ldots ^ Breadcrumb_{n} ^ \text{Specification} \Rightarrow \text{Requirement} \]
- tracability: trail of breadcrumbs
- identify unused phenomena
- handle general topologies
- formalize use of frame concern
- local patterns replace global patterns
Difficulties

- systematic not automatic (inescapable)
- readability, implementability, consistency
- which breadcrumb/rewrite?
- which push? split?
- get stuck later on?
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Cartoon of Big Example
Future Work

- patterns for local steps, concurrent steps
- proton therapy case study / safety case
- example/error progression
Related Work on Problem Frames

- Jackson, Zave (1995) turnstyle example
- Jackson (2001) problem progression
- Rapanotti, Hall, Li (2006) causal reasoning
- Hall, Rapanotti (2006) requirement progression
Problem Frames

- problem-oriented descriptions
- phenomena: observable
- domains: collections of phenomena
- requirement references phenomena
- machine controls phenomena to enforce requirement
- specification references controlled phenomena
Typical Transformation

- need to constrain $p_1$ instead of $p_2$
Typical Transformation

- need to constrain p1 instead of p2
- **add** a breadcrumb assumption relating p1 and p2
Typical Transformation

- need to constrain p1 instead of p2
- **add** a breadcrumb assumption relating p1 and p2
- **rephrase** the goal to reference p1 instead of p2 such that

\[ \text{Breadcrumb} \land \text{Rephrased Goal} \implies \text{Prior Goal} \]
Typical Transformation

• need to constrain p1 instead of p2

• **add** a breadcrumb assumption relating p1 and p2

• **rephrase** the goal to reference p1 instead of p2 such that

  \[\text{Breadcrumbs} \uparrow \text{Rephrased Goal} \Rightarrow \text{Prior Goal}\]

• **push** the goal towards the machine