

The slide features a decorative arrangement of six circles. Three light purple circles are positioned in the upper half, and three are in the lower half. The top row consists of an empty circle on the left, a solid circle in the middle, and another solid circle on the right. The bottom row consists of a solid circle on the left, a solid circle in the middle, and an empty circle on the right. The text is overlaid on these circles.

Name Consistency Checker in ATOM3

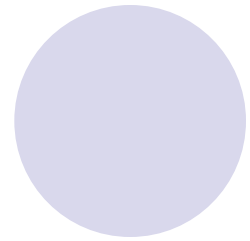
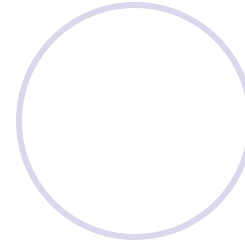
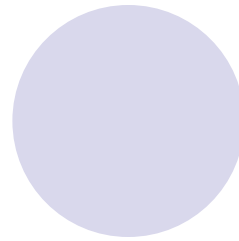
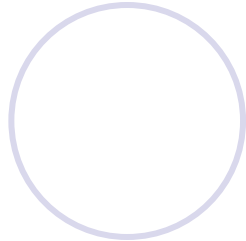
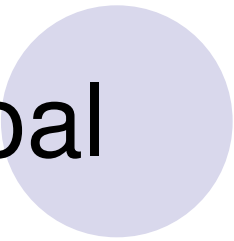
Presented by Victoria Yang

A decorative graphic at the top of the slide consists of two groups of circles. The first group on the left has a solid light purple circle on the left and an empty light purple circle outline on the right. The second group on the right has a solid light purple circle on the left, an empty light purple circle outline in the middle, and a solid light purple circle on the right.

Overview

- Main constraint
- Constrain Checker (CC) revisited
- Implementation in ATOM3
- Demo
- Future Work

Goal



- Enforce name consistency over elements in a model.
- Petri Net Example.

Recall: The Constraint Checker (CC)

1. Map consistency constraint to checking rule:
 - *if* **Expression** *then* **Action**
2. Apply checking rule to model.

The Constraint



- UML Well-formedness rule:

- *If a contained element, which is not an Association or Generalization, has a name then the name must be unique in the Namespace*

Generalization of Rule

- Well-formedness Rule:

- *No instantiation of the same meta element may have the same name*

Petri Net Name Constraints

The title is centered at the top of the slide. Behind the text are five circles arranged horizontally. From left to right, they are: a solid light purple circle, an empty white circle with a light purple outline, a solid light purple circle, an empty white circle with a light purple outline, and a solid light purple circle.

- 1. No two places may have the same name within a model.*
- 2. No two transitions may have the same name within a model.*

Map Name Constraint to Checking Rule

- Checking rule:
 - If *two instantiations of the same meta element have the same name* then *they should be the same instantiation.*



Problem with Checking Rule

- Checking rule implies should merge similarly name places and transitions into one.
- Problem:
 - if modeler intended for separate entities, merger will cause loss of info.
- Solution:
 - rename one entities instead of merging.



Alteration of Checking Rule

- New Checking Rule:

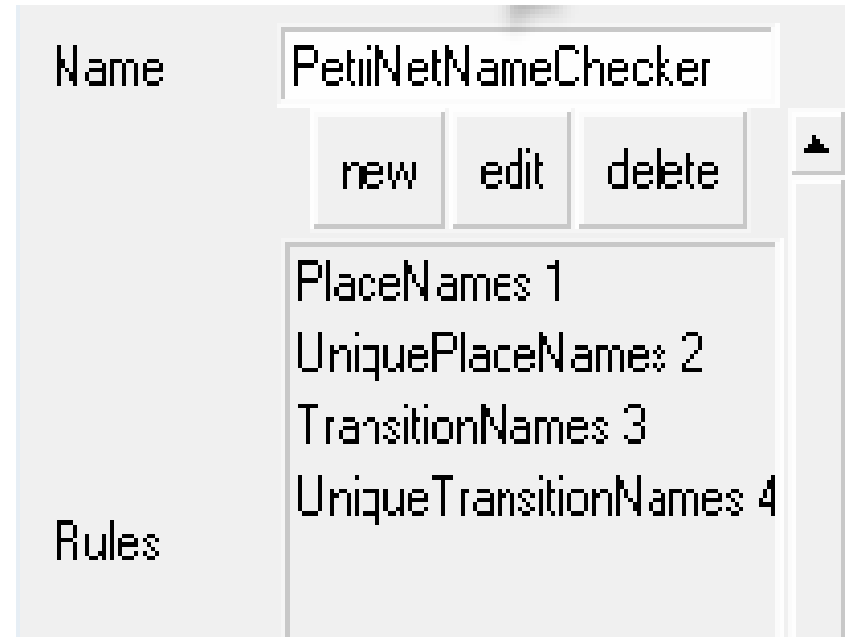
- If *two instantiations of the same meta element has the same name* then *rename one of the instantiations*

Petri Net Checking Rules

1. If *two places have the same name within the model*, then *rename a place of them*
2. If *two transitions have the same name within the model*, then *rename a transition of them*

CC Implementation in ATOM3

- Implement constraint checking rules as graph grammars in ATOM3's graph rewriting system.





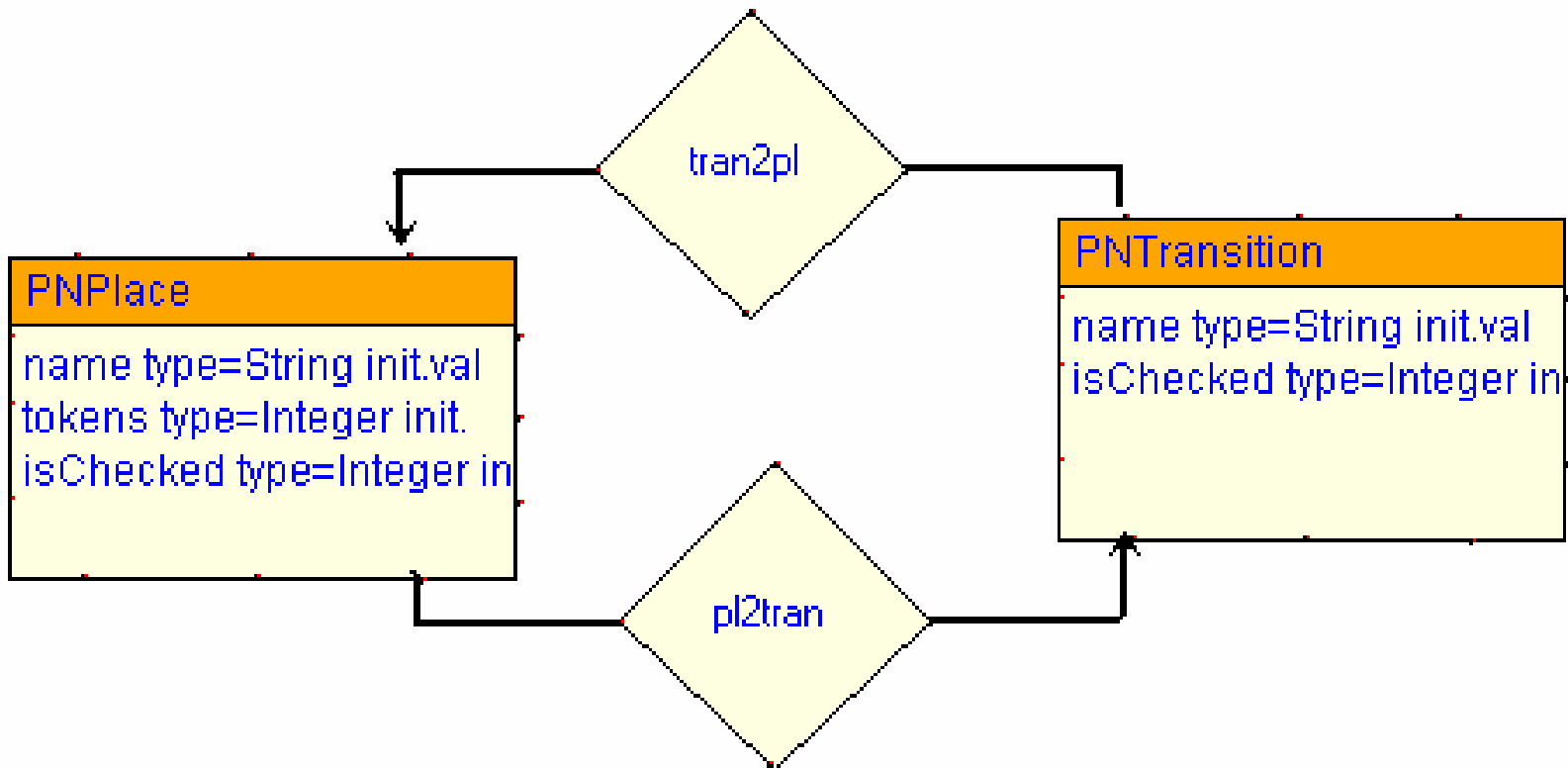
Implementation Issues

- Problems:

- Need to know whether entity has been checked.
- Need to trace changes made (which entities were renamed).

Solution: Use Stereotypes

- Extend metamodel with a tagged value.



The Tagged Values



- Implemented as an integer attribute of Place and Transition for Petri Net.
- Tagged Values
 - (isChecked, value)
 - Value = 0: Initial
 - Value = 1: Name is unique
 - Value = 2: Name is not unique

Graph Rewriting Rules for CC

The title is centered at the top of the slide. It is surrounded by five circles of varying shades of purple and lavender. The circles are arranged in a horizontal line, with some overlapping the text. The first circle is solid purple and overlaps the 'G'. The second is a light purple outline and overlaps the 'R'. The third is solid purple and overlaps the 'R'. The fourth is a light purple outline and overlaps the 'W'. The fifth is solid purple and overlaps the 'R'.

- Preprocessing
- Rule 1: Place Names
 - Add all names of places to the global name list
 - Set their tagged values to 1

Graph Rewriting Rules for CC (2)

- Apply checking rules for places in CC
- Rule 2: Unique Place Names
 - Look up duplicated names in the global name list
 - Rename non-unique names until no more duplicated names are found
 - Add new name to global name list, remove its old name and set their tagged value to 2

Graph Rewriting Rules for CC (3)

- Rules for transition namespace
 - Similar to rules for place namespace
- Rule 3 : Preprocessing
- Rule 4 : Apply checking rules for transitions



Future Work with ATOM3

- Automation of extending meta element
with tags
- Get user input to force action

References



- Jean Louis Sourrouille, Guy Caplat: *Checking UML Model Consistency*, Workshop on Consistency Problems in UML-based Software Development, 2002
- John Hendrik Haumann, Reiko Heckel, and Stefan Sauer: *Extended Model Relations with Graphical Consistency Conditions*, Workshop on Consistency Problems in UML-based Software Development, 2002
- WenQian Liu, Steve Easterbrook and John Mylopoulos: *Rule-based Detection of Inconsistency in UML Models, Conditions*, Workshop on Consistency Problems in UML-based Software Development, 2002
- UML Semantics: <http://www.inf.int-evry.fr/COURS/UML/semantics/>, INT - Département INFormatique, France