

**2007 Bellairs
CAMPaM Workshop**

A Study of Model Transformation Technologies: Reconciling TGGs with QVT

Joel Greenyer

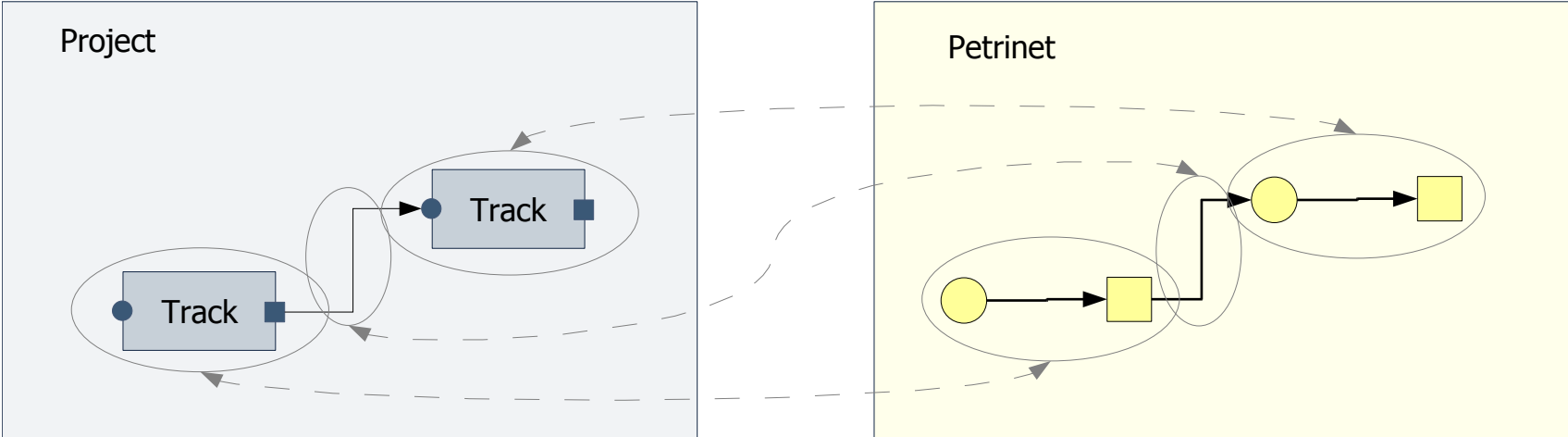
Diploma Thesis supervised by Dr. Ekkart Kindler and Robert Wagner



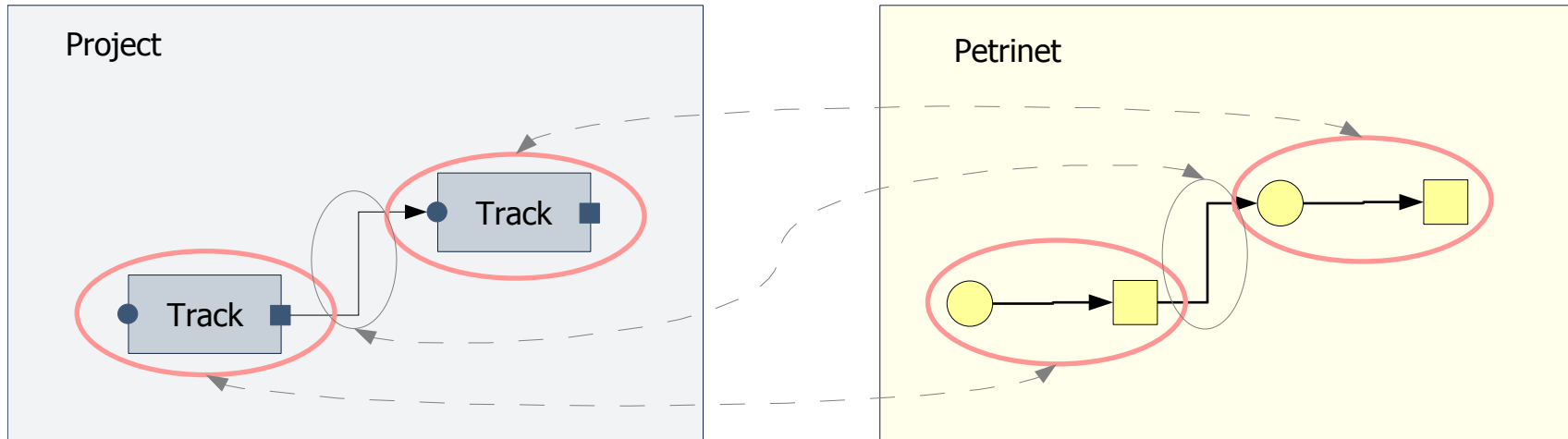
UNIVERSITÄT PADERBORN
Institut für Informatik – FG Softwaretechnik

Software Engineering Group

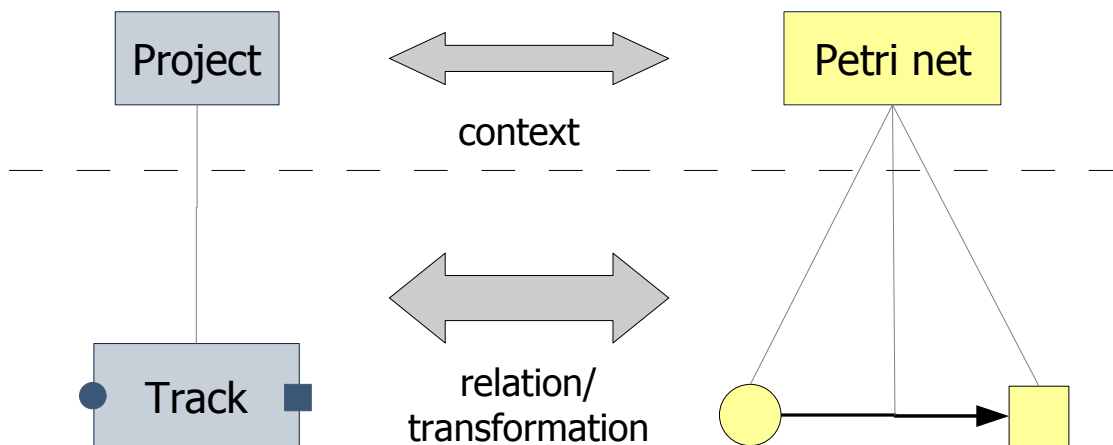
Example: ComponentTools To Petrinet



Example: ComponentTools To Petrinet

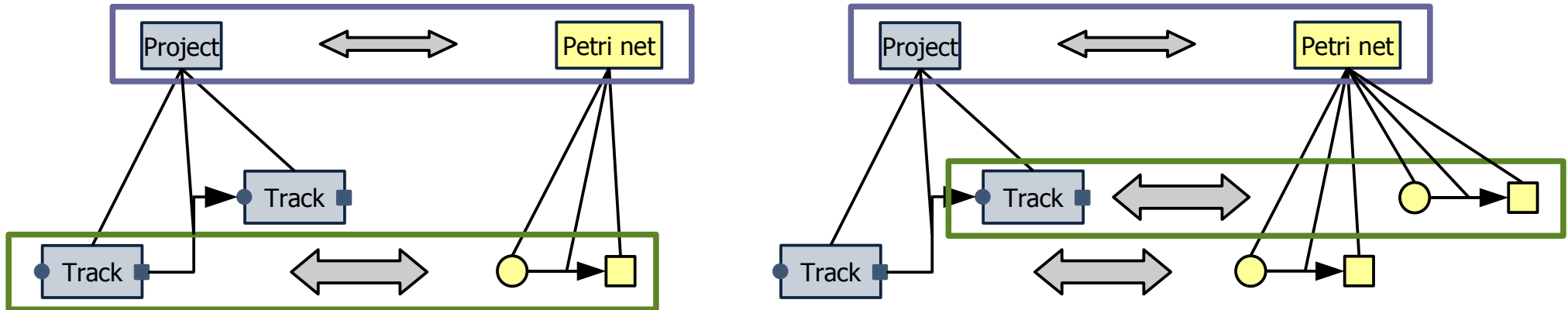


- Example Rule: TrackToPlaceArcTransition

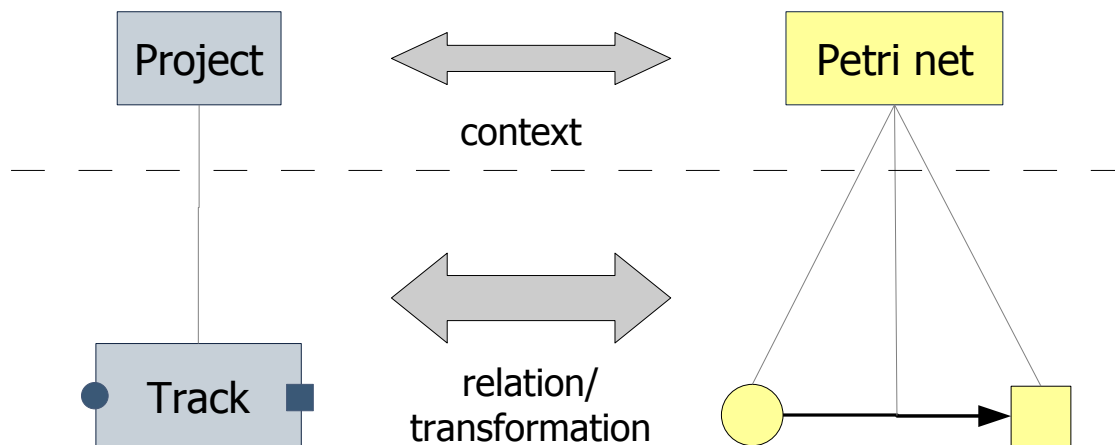


- Declarative
- Relational
- Graph-pattern-based

Example: ComponentTools To Petrinet



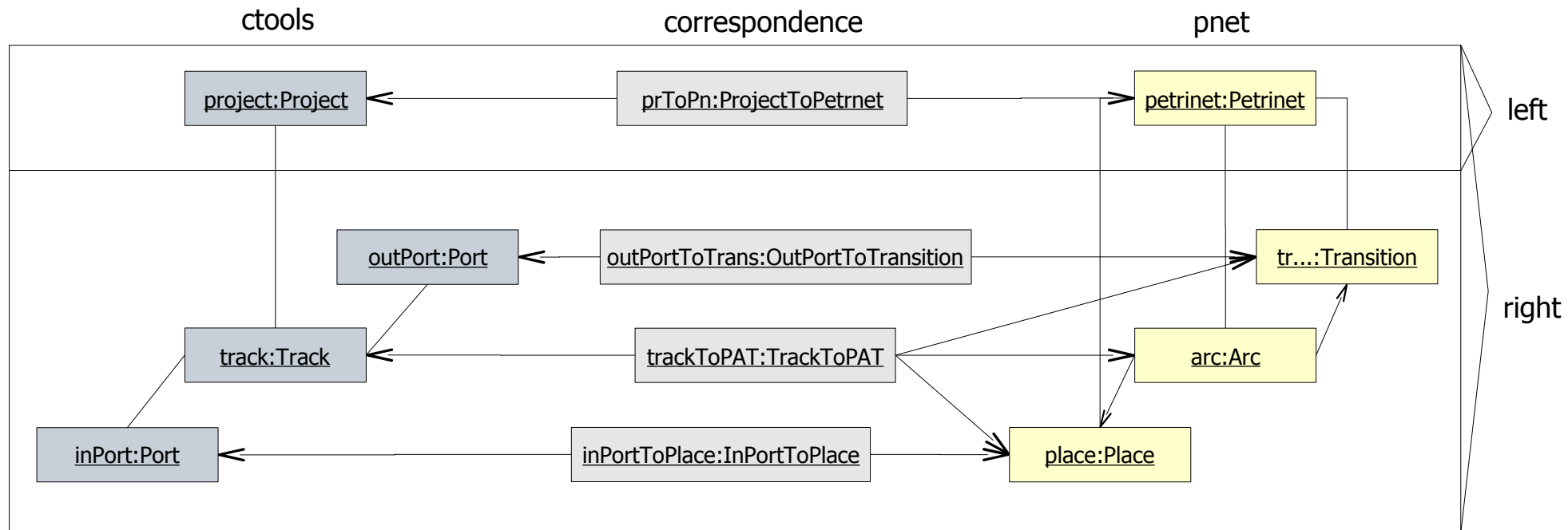
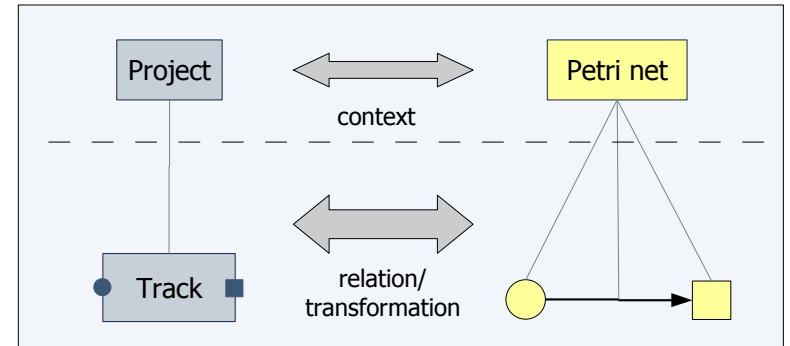
- Example Rule: TrackToPlaceArcTransition



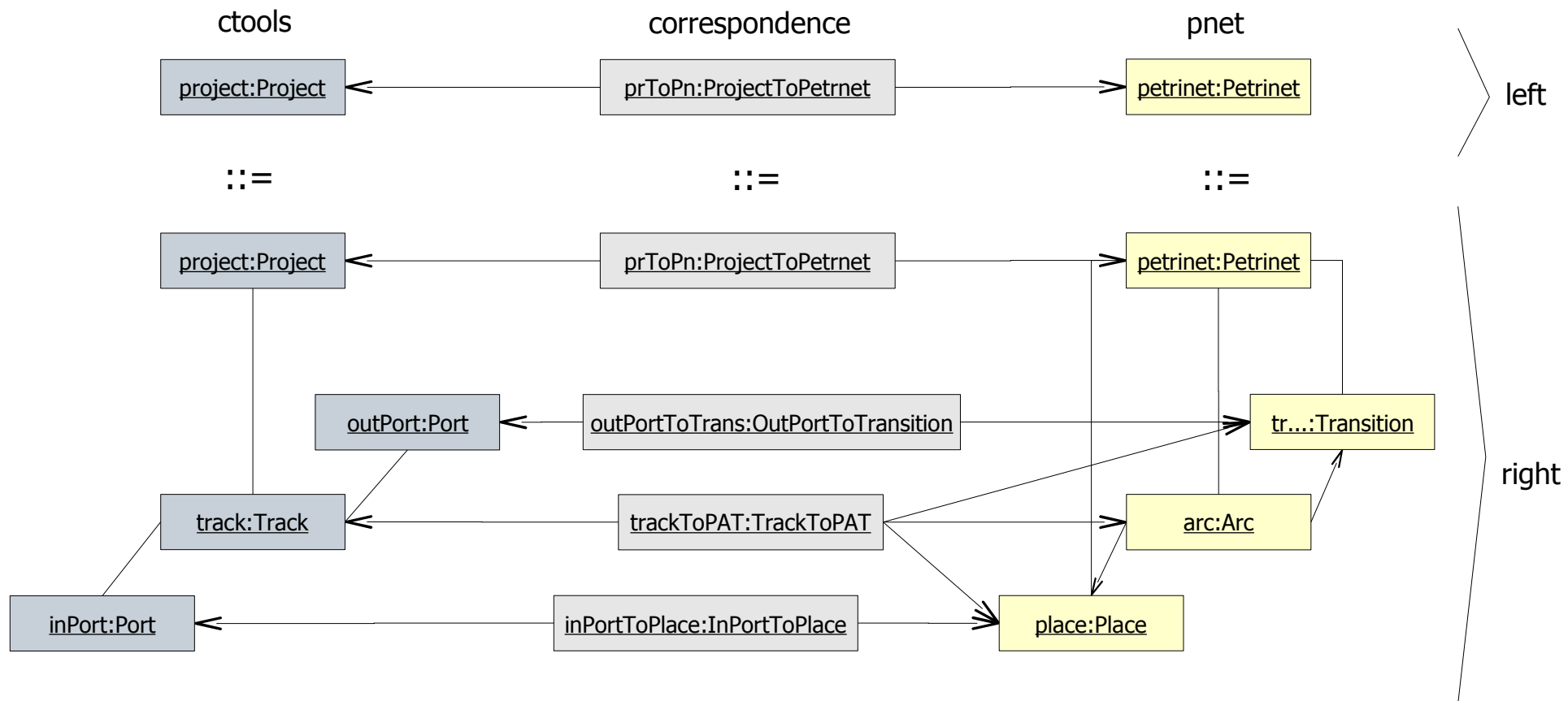
- Declarative
- Relational
- Graph-pattern-based

TGG Rule TrackToPlaceArcTransition

- The corresponding TGG rule:

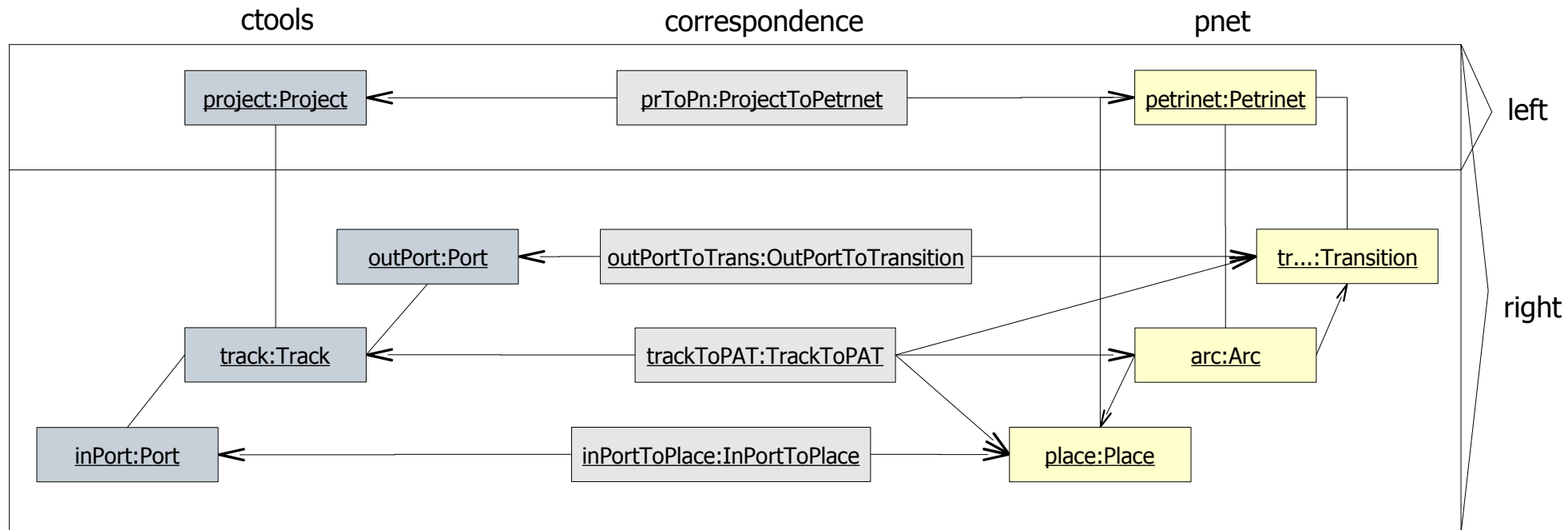
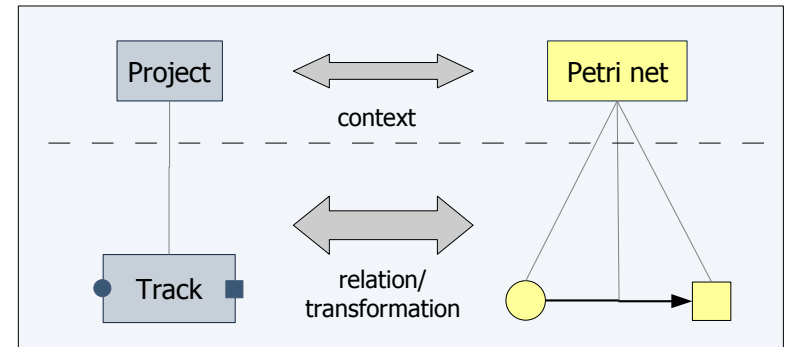


TGG Rule TrackToPlaceArcTransition



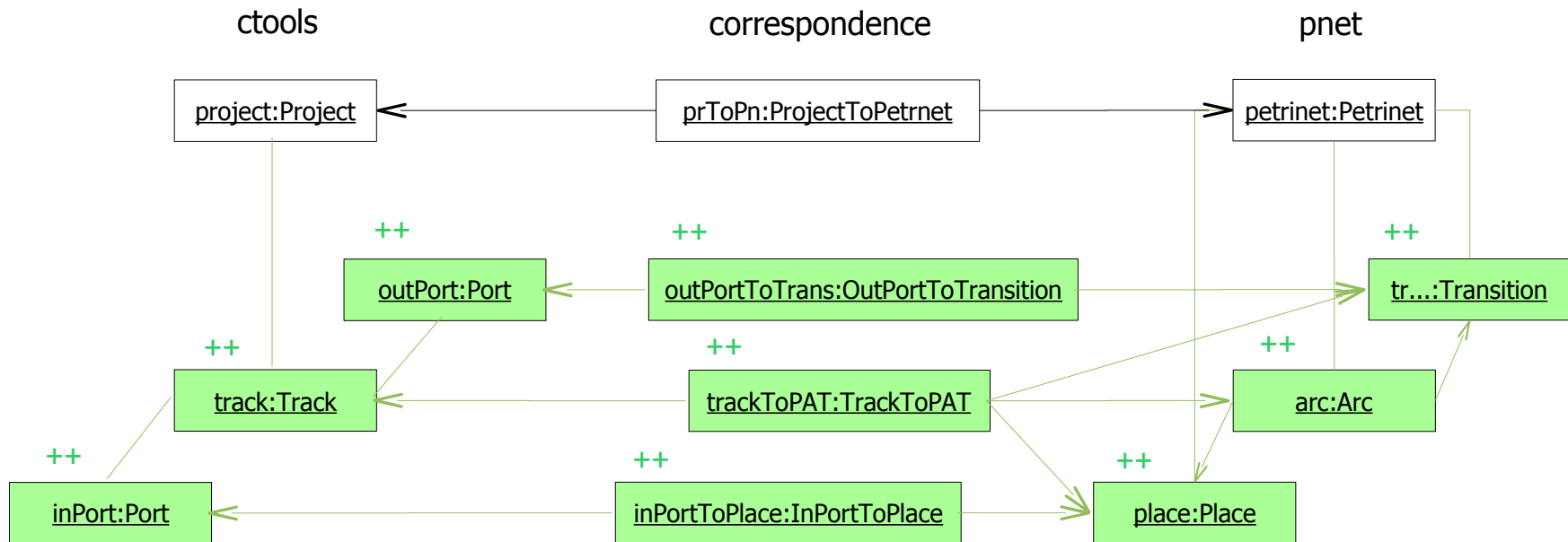
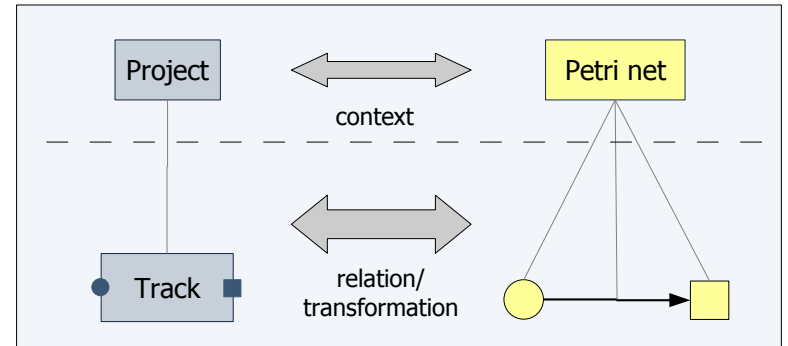
TGG Rule TrackToPlaceArcTransition

- The corresponding TGG rule:

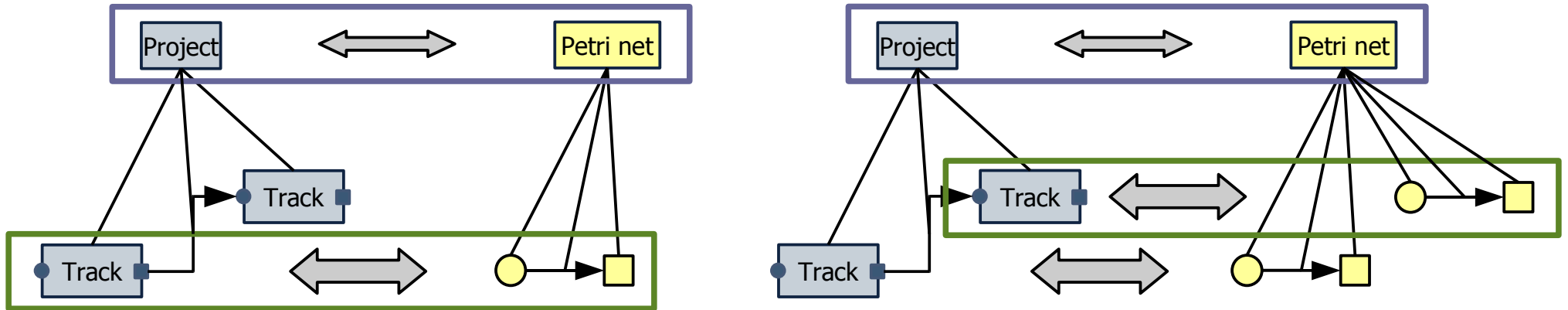


TGG Rule TrackToPlaceArcTransition

- The corresponding TGG rule:



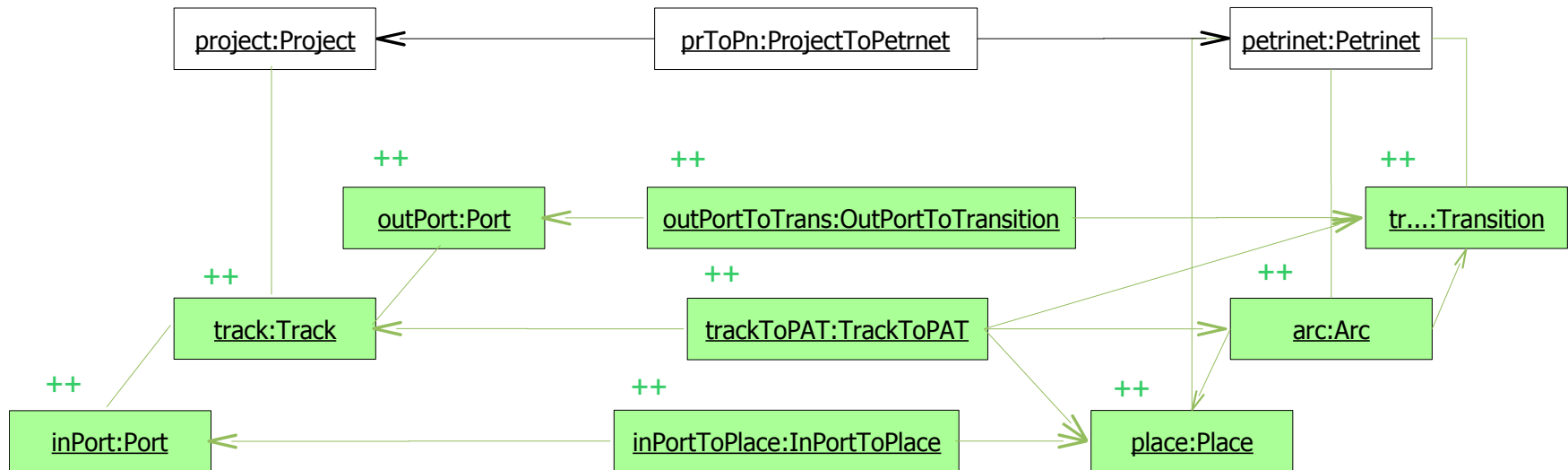
TGG Rule TrackToPlaceArcTransition



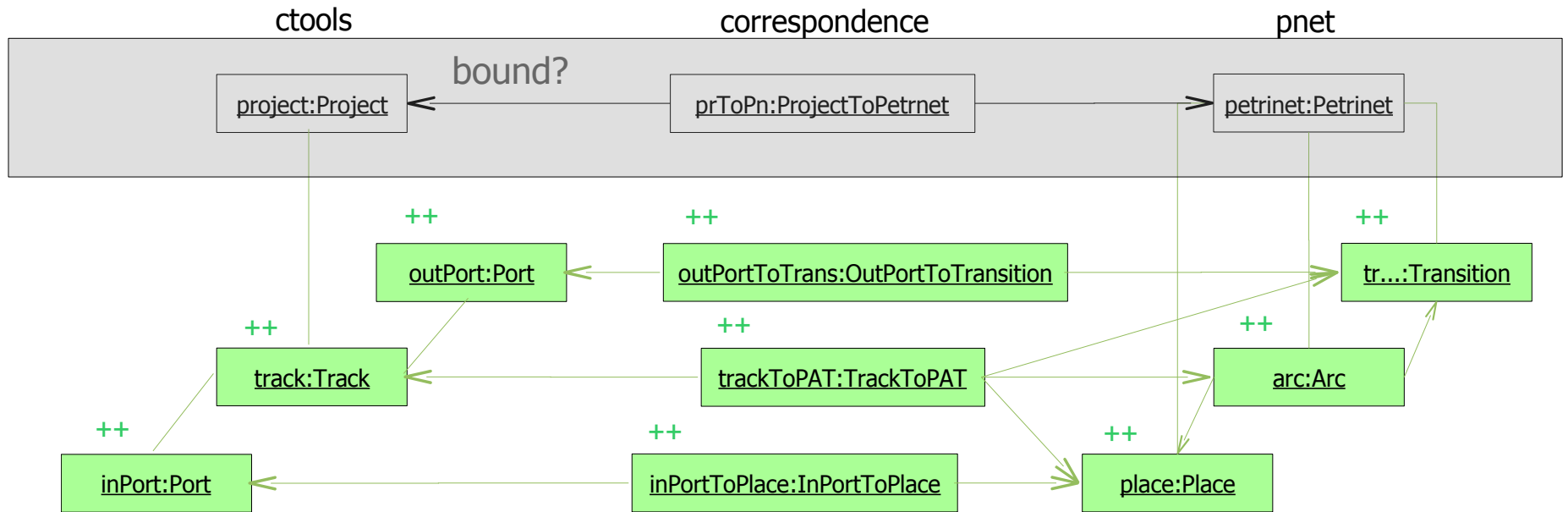
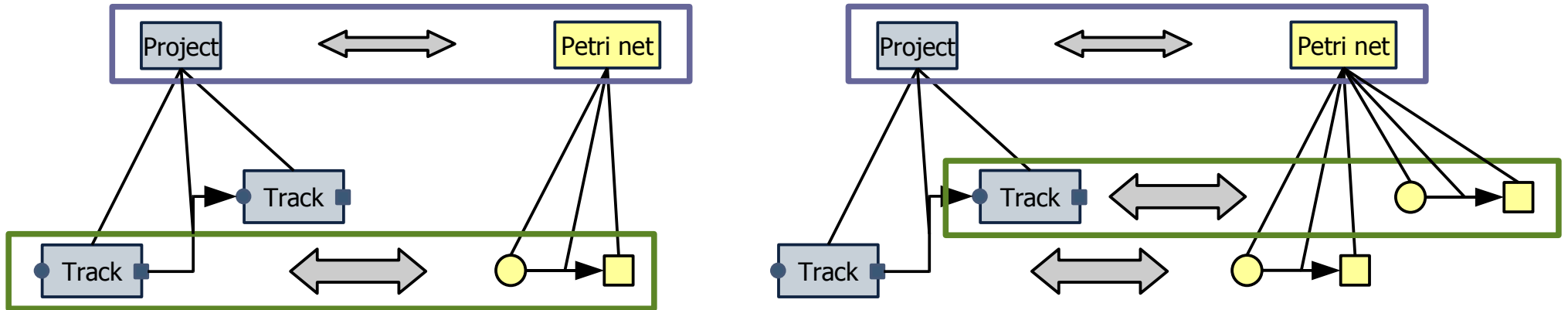
ctools

correspondence

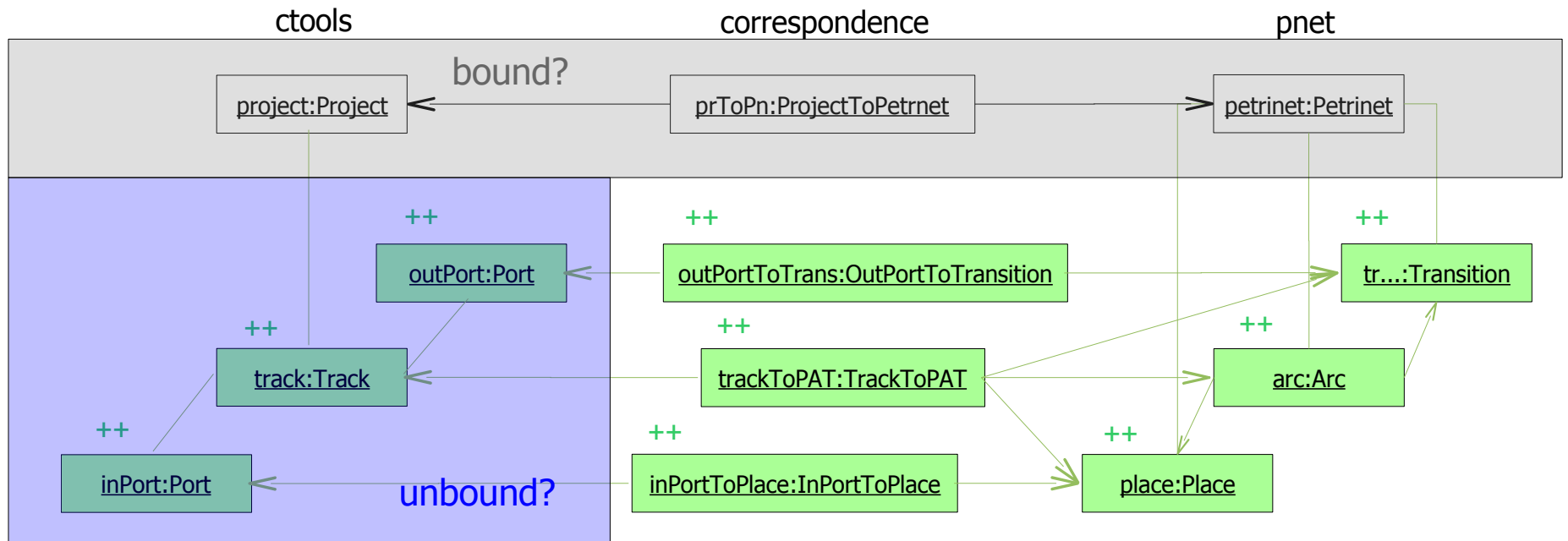
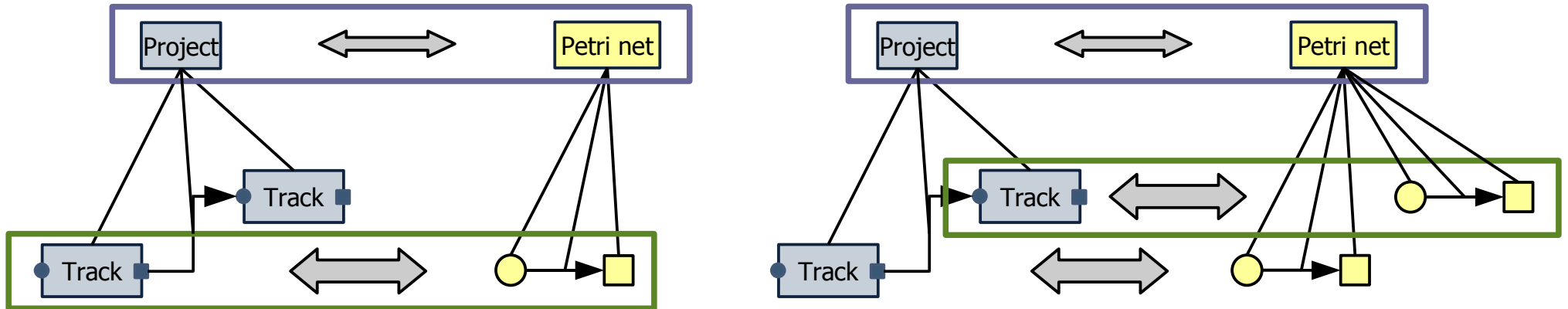
pnet



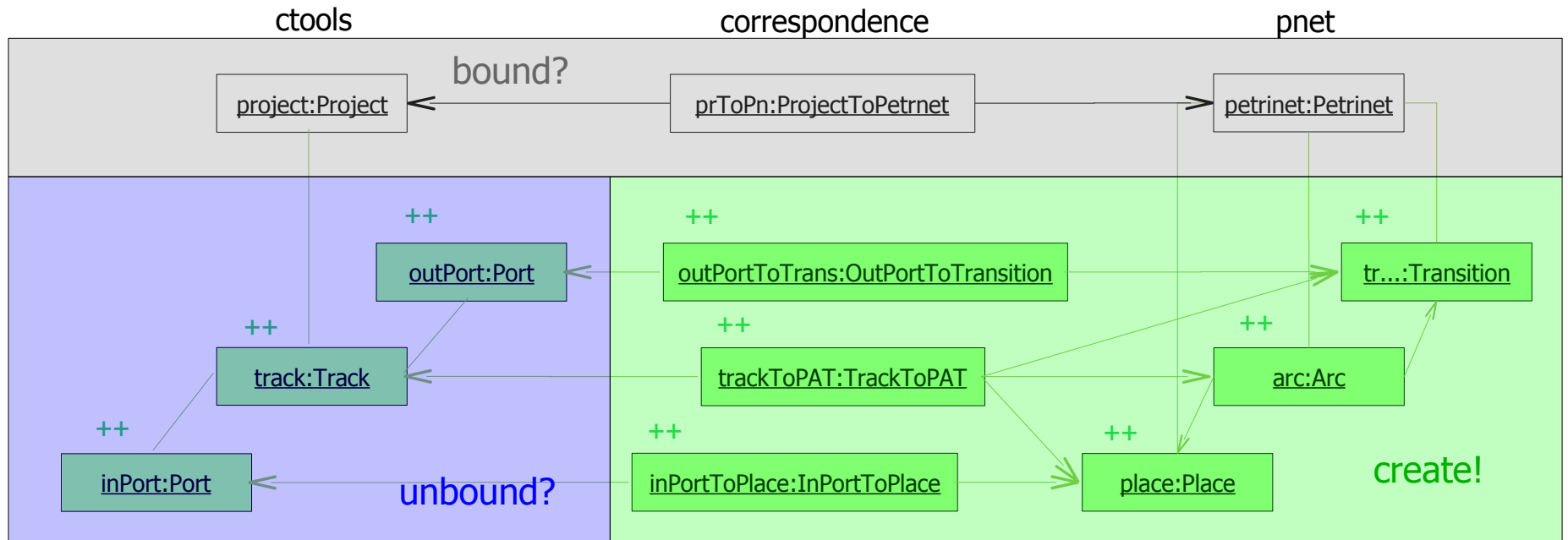
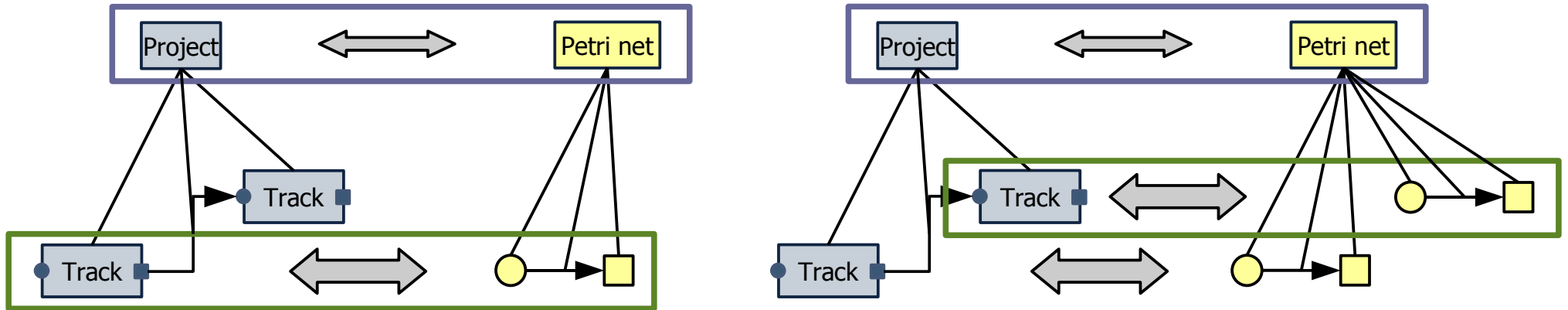
TGG Rule TrackToPlaceArcTransition



TGG Rule TrackToPlaceArcTransition

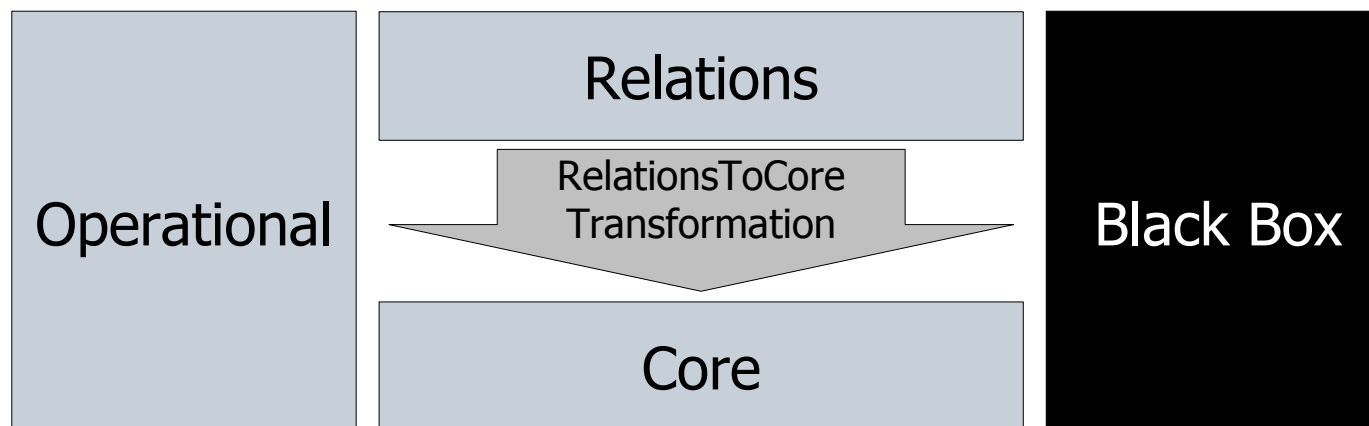


TGG Rule TrackToPlaceArcTransition



QVT Overview

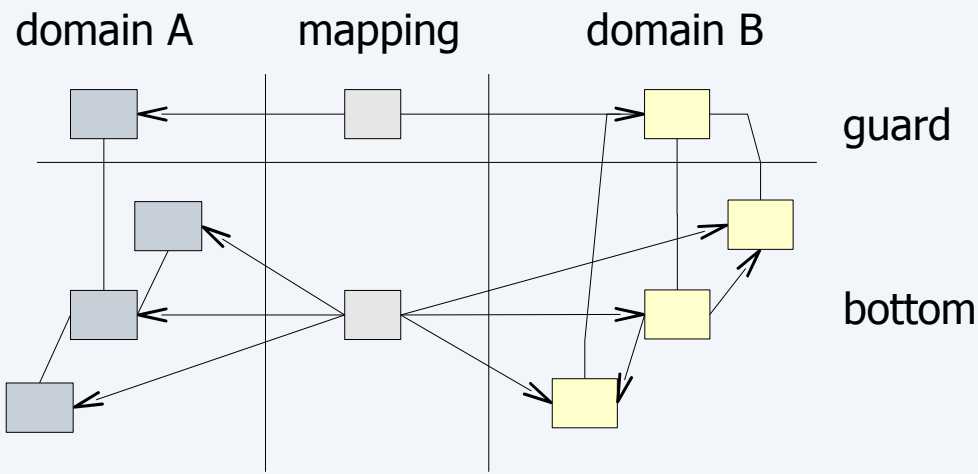
- Model Transformation Approaches in MDD and MDA
 - Many techniques, different strengths and weaknesses
- OMG's QVT (Query/Views/Transformations)^[2]
 - Result of selecting and merging different approaches
 - Combination of declarative and imperative languages
 - Focus on declarative languages



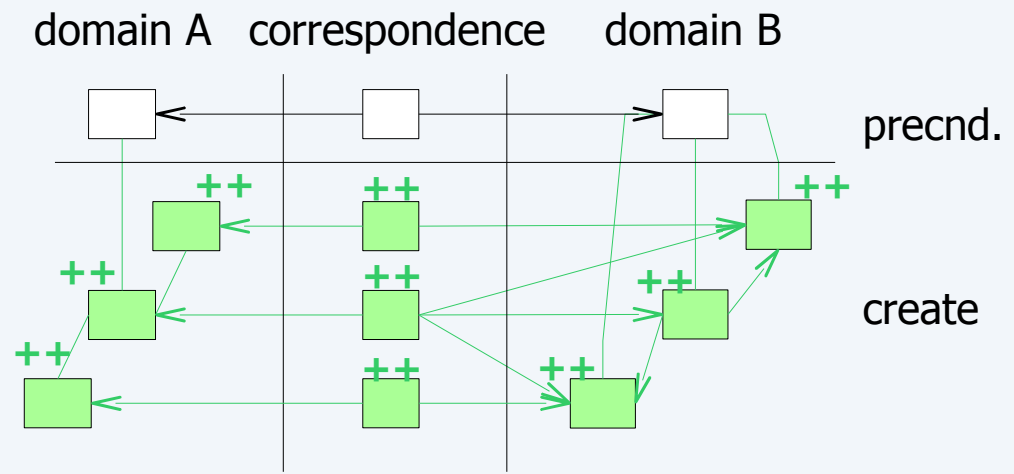
QVT-Core vs. TGGs – Short Comparison

- Structurally similar rules

QVT-Core*



TGG



- Graph structure: Variables and OCL-Predicates
- One *Trace Class*

- Graph structure: Nodes and Edges
- Multiple *Correspondence Nodes*

*:QVT-Core does not specify a graphical notation

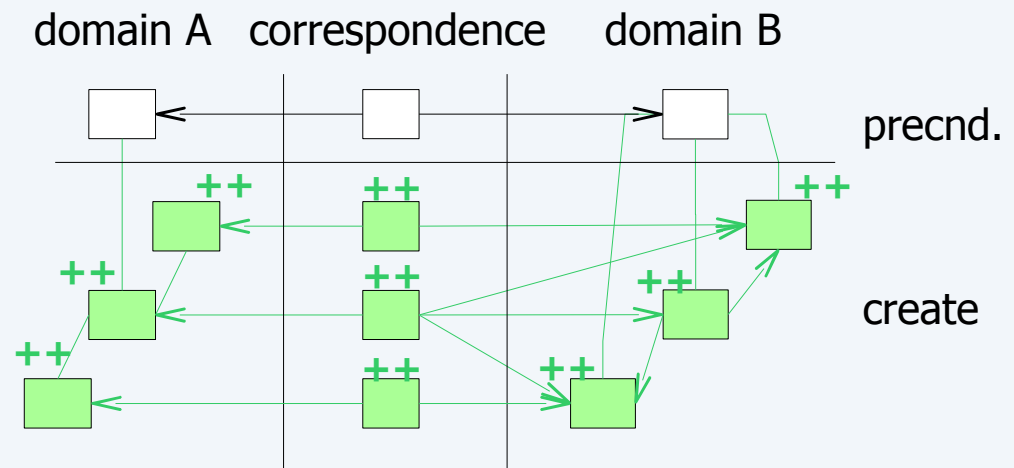
QVT-Core vs. TGGs – Short Comparison

- Structurally similar rules

QVT-Core

```
map TrackToPlaceArcTransition{  
  
  check ctools(project:Project){  
    track:Track, portIn:Port|  
    track.project = project;  
    track.port = portIn;  
    track.port = portOut;  
  }  
  
  check enforce pnet(petrinet:PetriNet){  
    realize place:Place, ...  
  }  
}
```

TGG

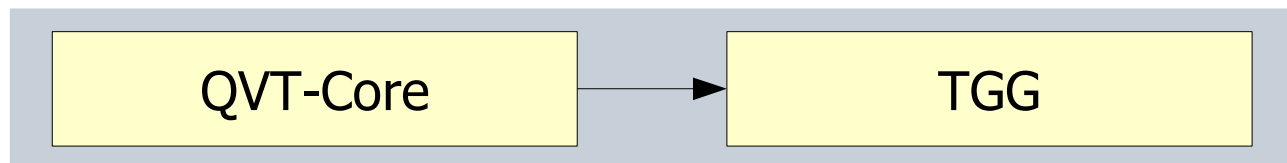


- Graph structure: Variables and OCL-Predicates
- One *Trace Class*

- Graph structure: Nodes and Edges
- Multiple *Correspondence Nodes*

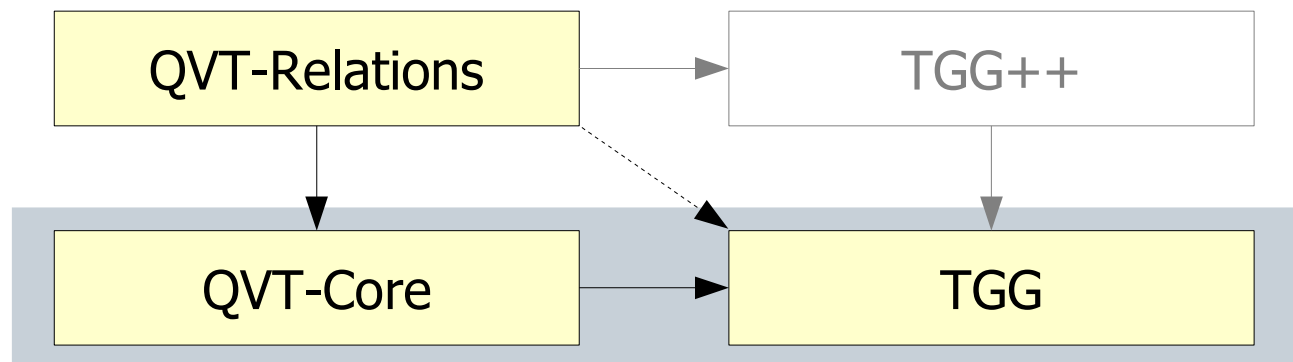
Mapping QVT to TGGs

- Main focus: Mapping QVT-Core to TGG
 - Realized through a TGG transformation
 - For fundamental pattern structure expressions



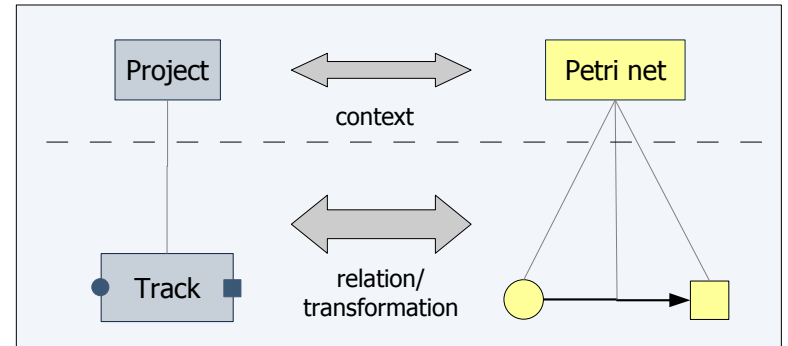
Mapping QVT to TGGs

- Main focus: Mapping QVT-Core to TGG
 - Realized through a TGG transformation
 - For fundamental pattern structure expressions

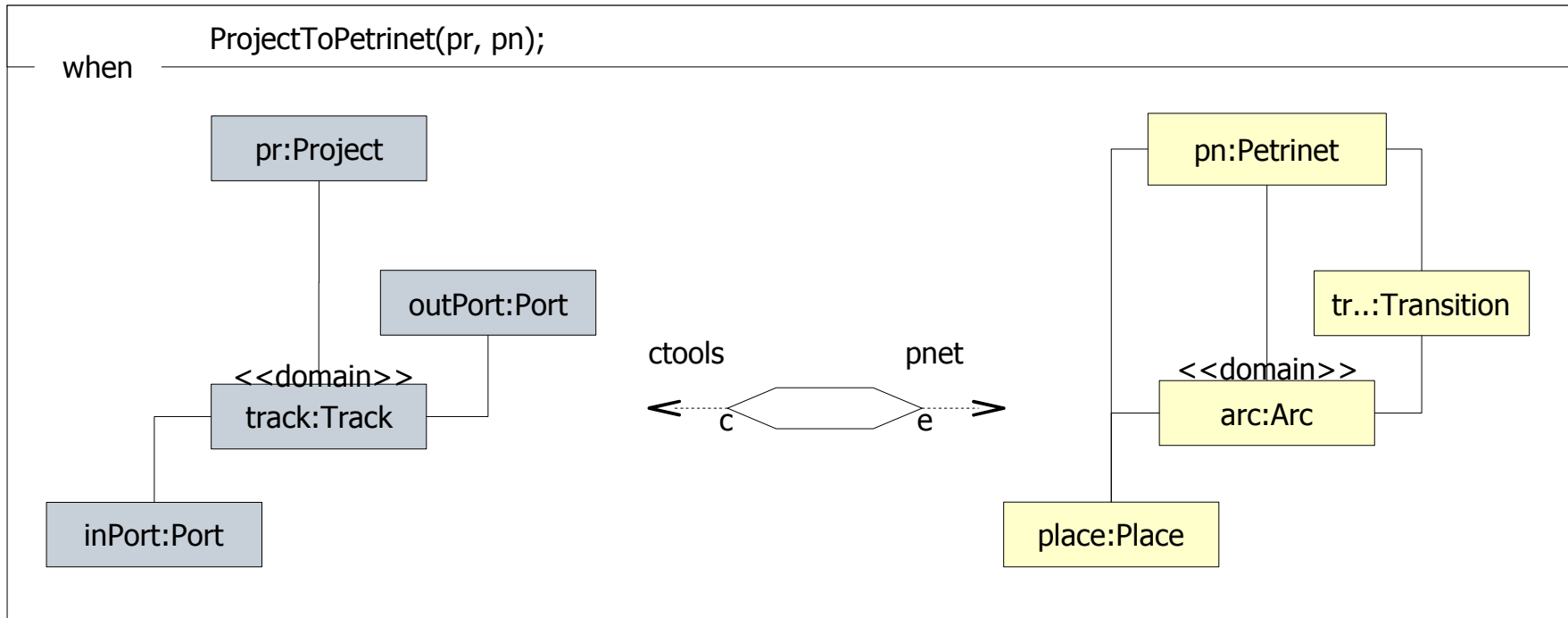


- Approach presented: QVT-Relations to TGG
- Learn from QVT: More usable “TGG++”?

QVT Relation TrackToPlaceArcTransition

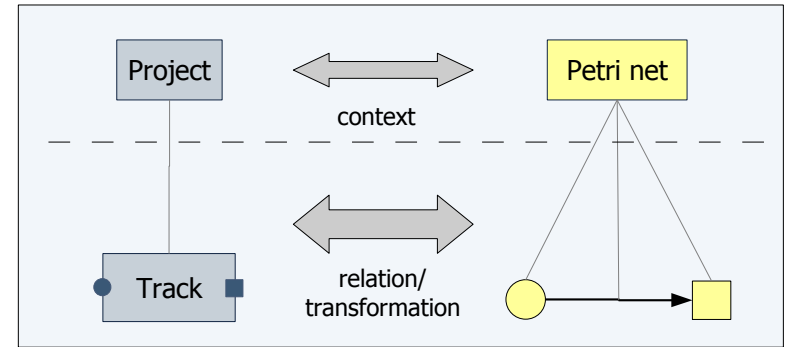
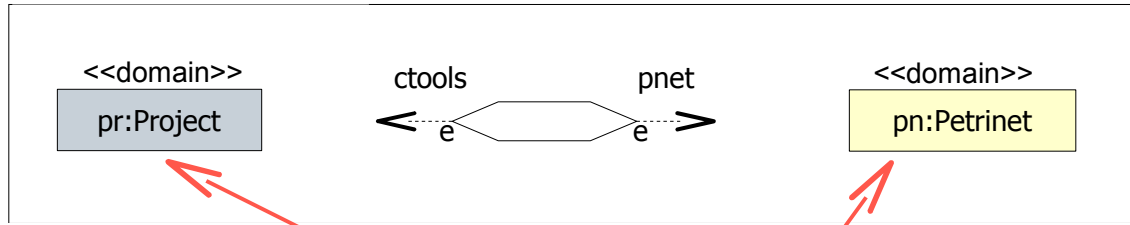


Relation TrackToPlaceArcTransition

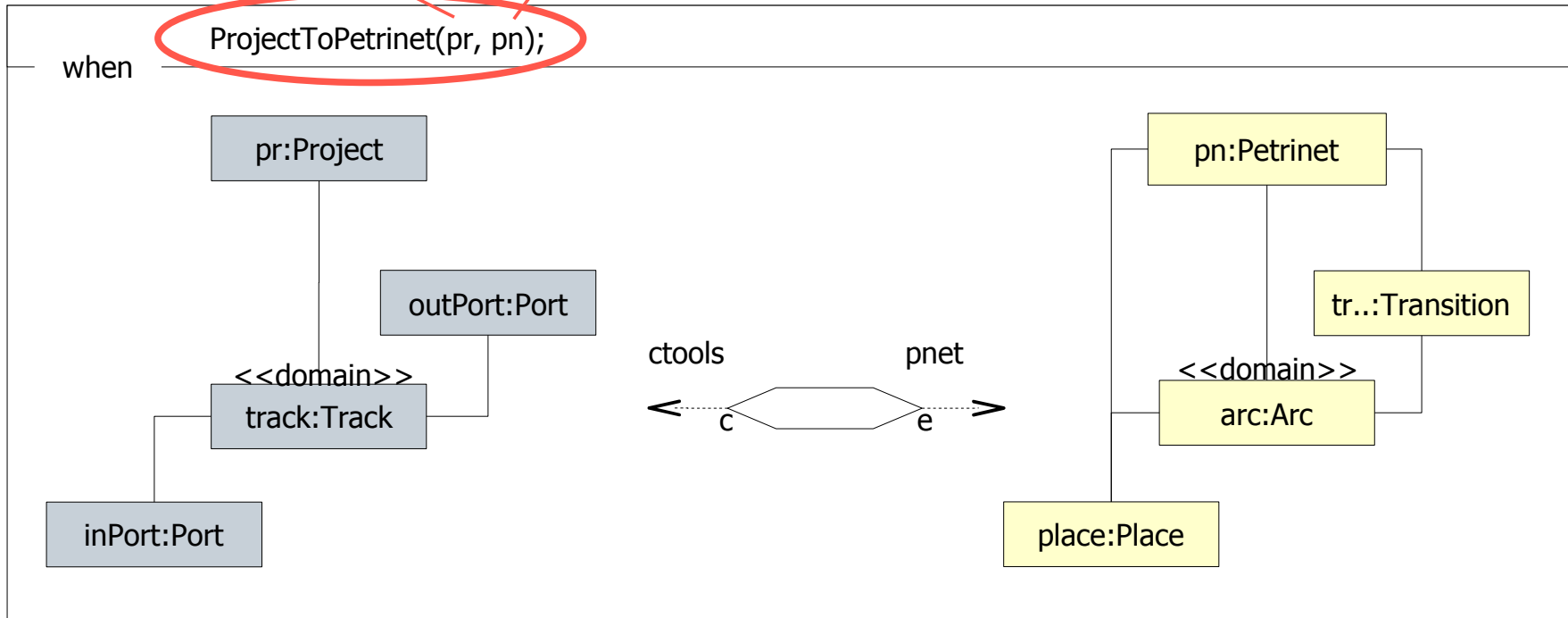


QVT Relation TrackToPlaceArcTransition

ProjectToPetrinet

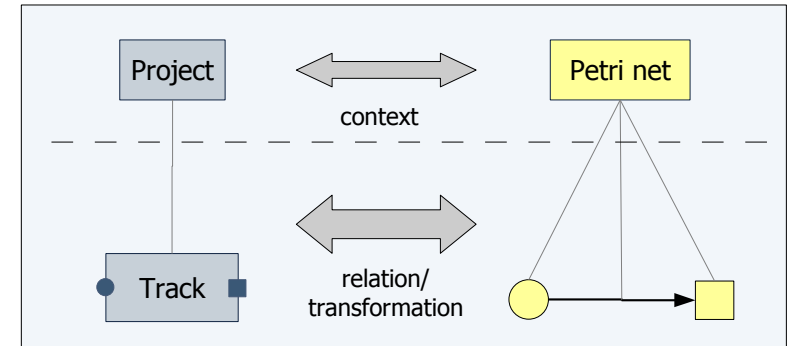
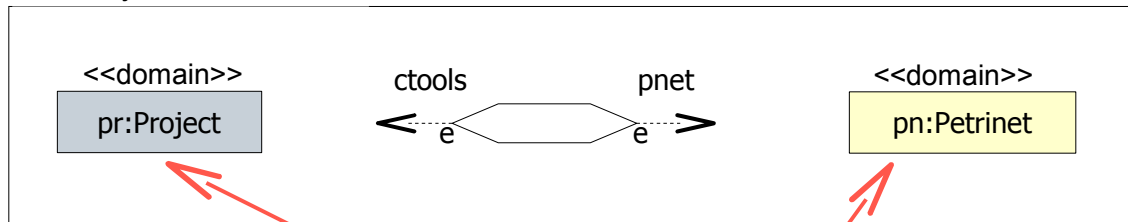


Relation TrackToPlaceArcTransition



QVT Relation TrackToPlaceArcTransition

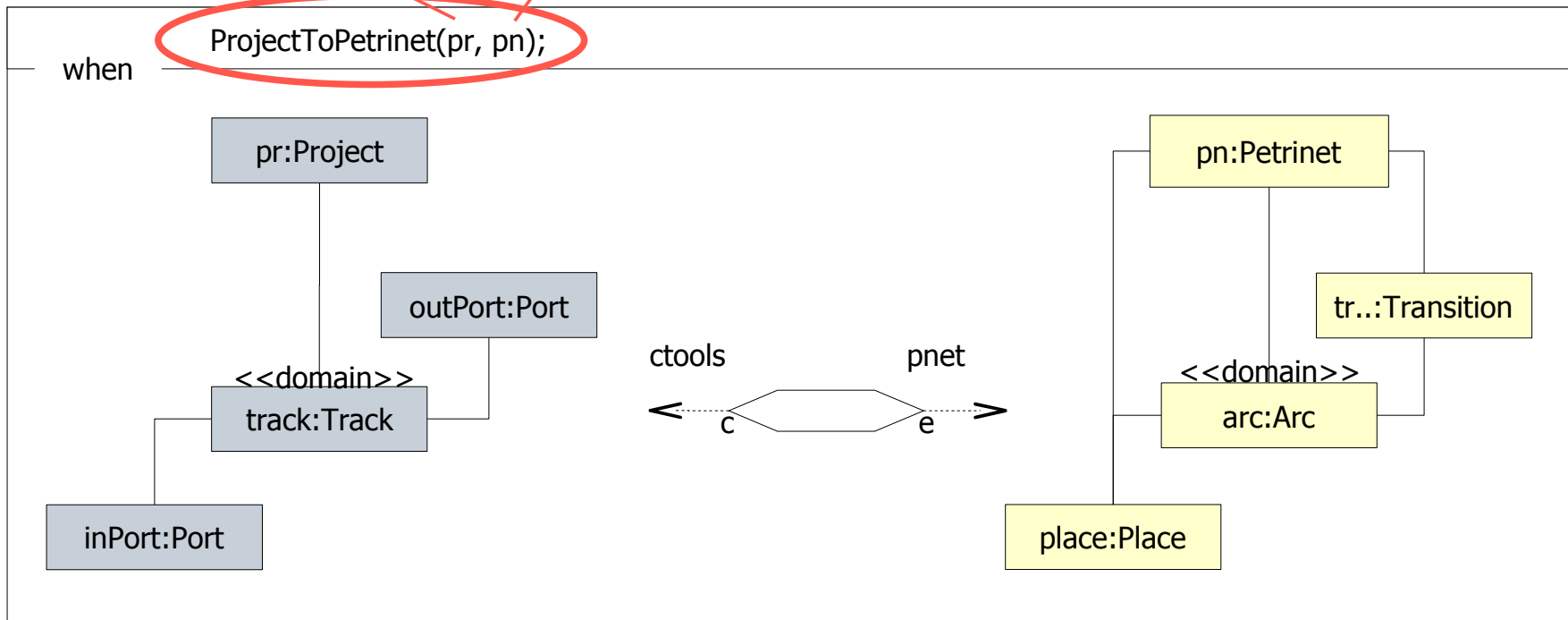
ProjectToPetrinet



Domain-Element:

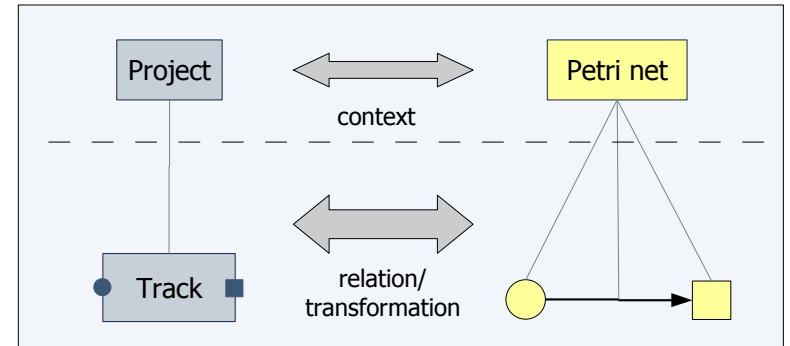
- "Root" of pattern
- Parameter for "relation calls"

Relation TrackToPlaceArcTransition

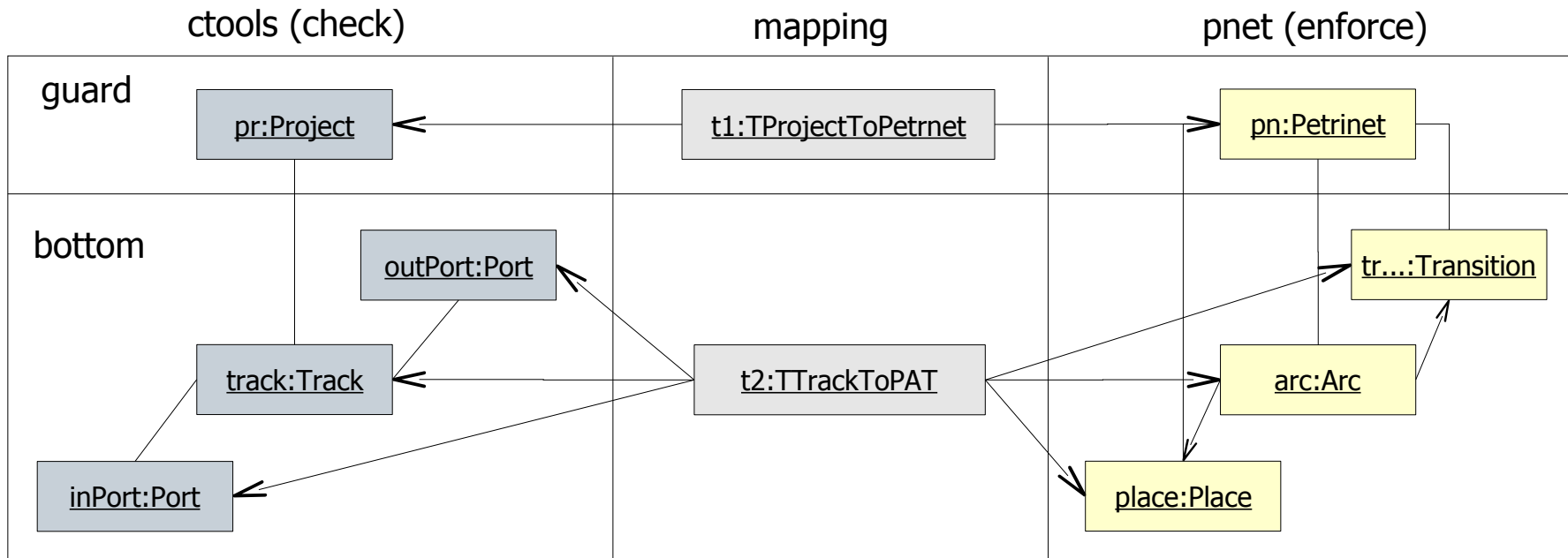


QVT-Core Mapping TrackToPlaceArcTransition

- Is transformed into a QVT-Core Mapping:

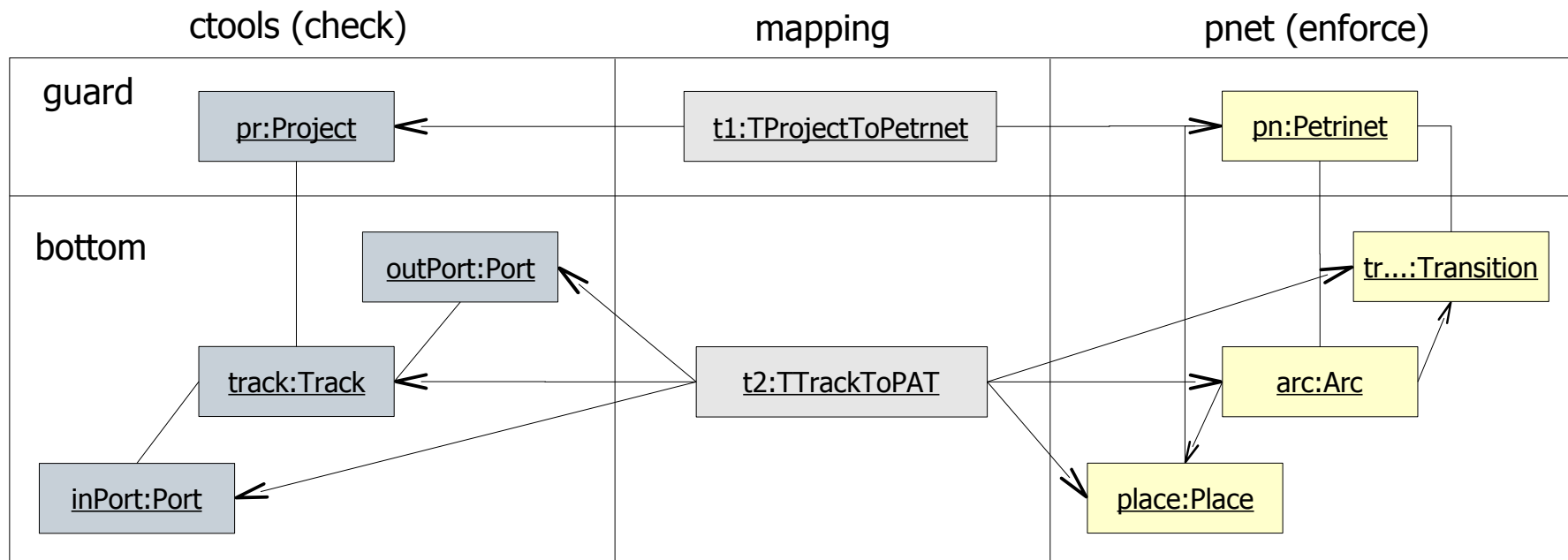


Mapping TrackToPlaceArcTransition



Semantics of a QVT-Core Mapping (fwd)

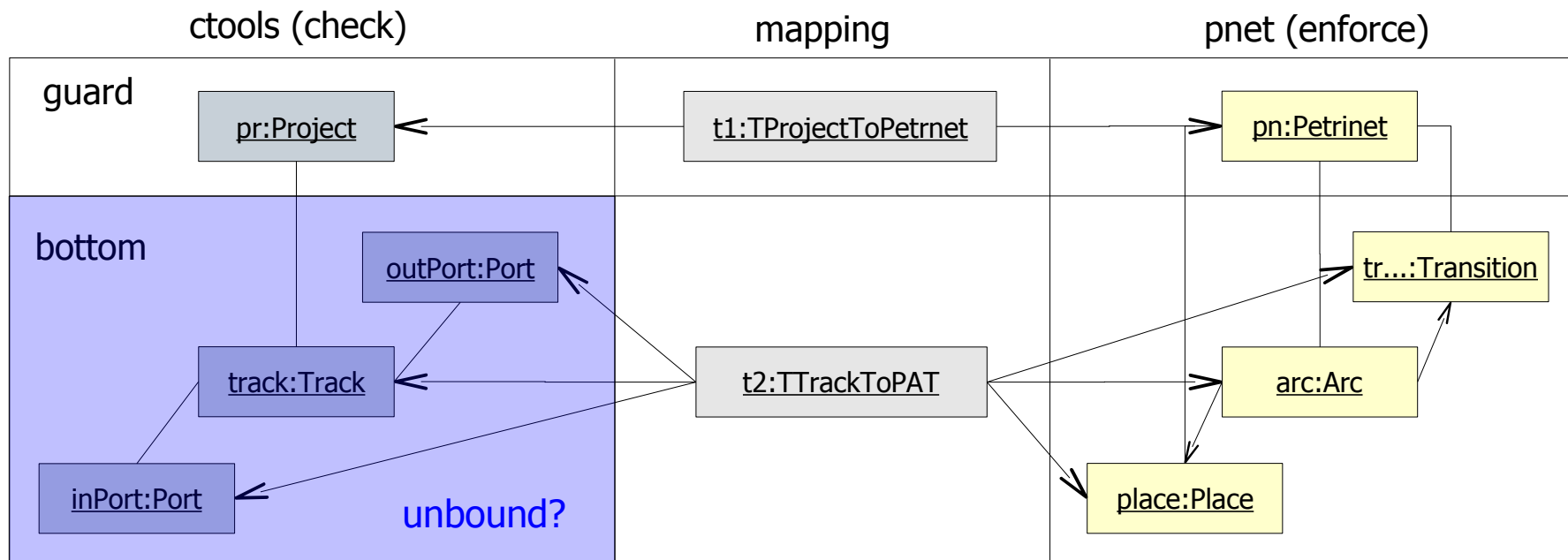
Mapping TrackToPlaceArcTransition



Semantics of a QVT-Core Mapping (fwd)

1) If Bottom Pattern found in unbound source model

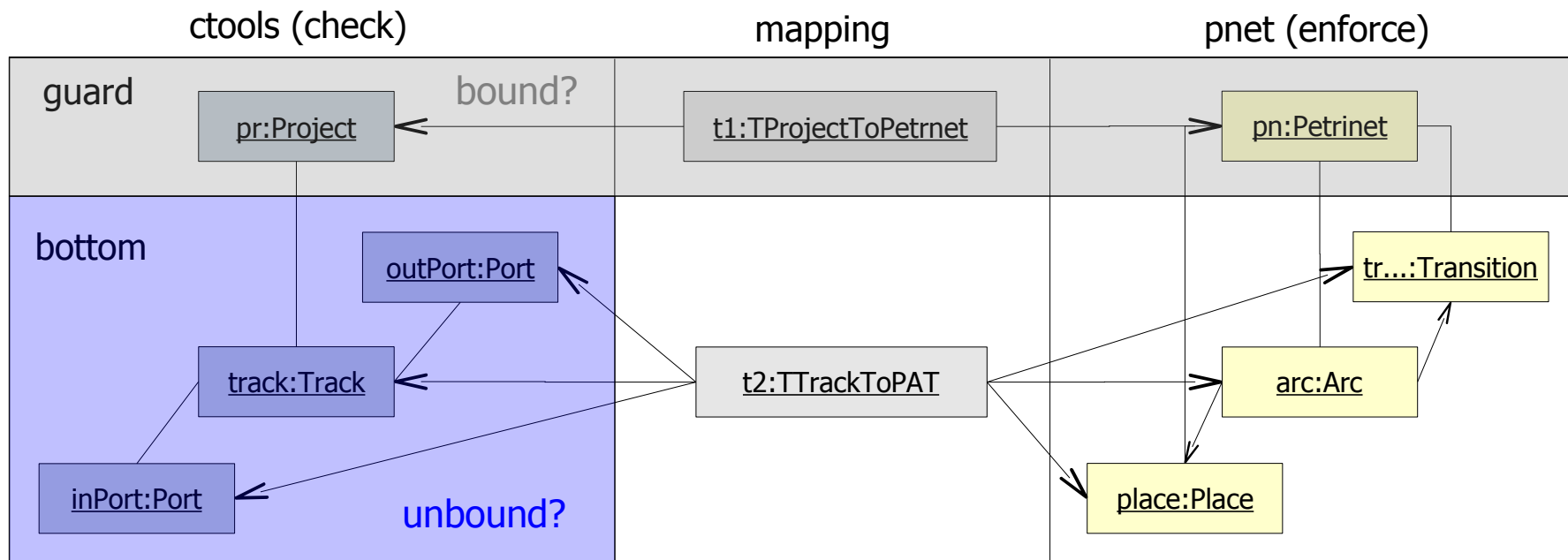
Mapping TrackToPlaceArcTransition



Semantics of a QVT-Core Mapping (fwd)

- 1) If Bottom Pattern found in unbound source model
- 2) & If Guard Pattern found in prev. bound model

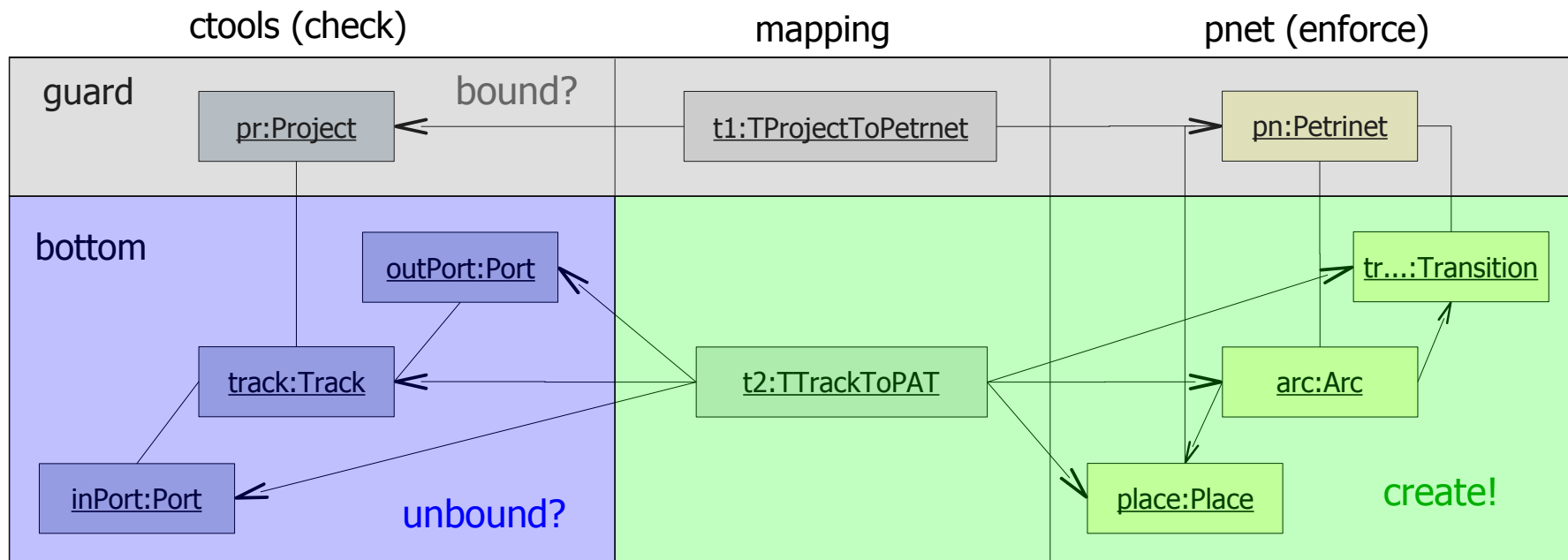
Mapping TrackToPlaceArcTransition



Semantics of a QVT-Core Mapping (fwd)

- 1) If Bottom Pattern found in unbound source model
- 2) & If Guard Pattern found in prev. bound model
- 3) Then create target model (and Trace Class)

Mapping TrackToPlaceArcTransition



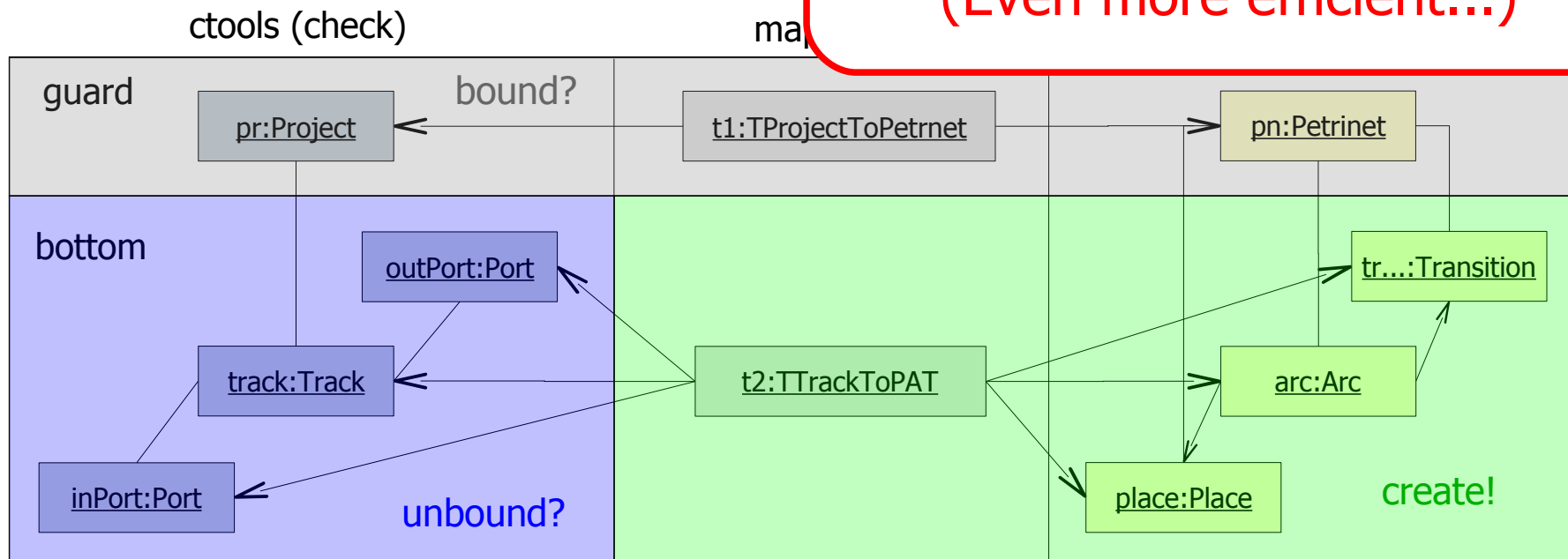
Semantics of a QVT-Core Mapping (fwd)

- 1) If Bottom Pattern found in unbound source model
- 2) & If Guard Pattern found in prev. bound model
- 3) Then create target model



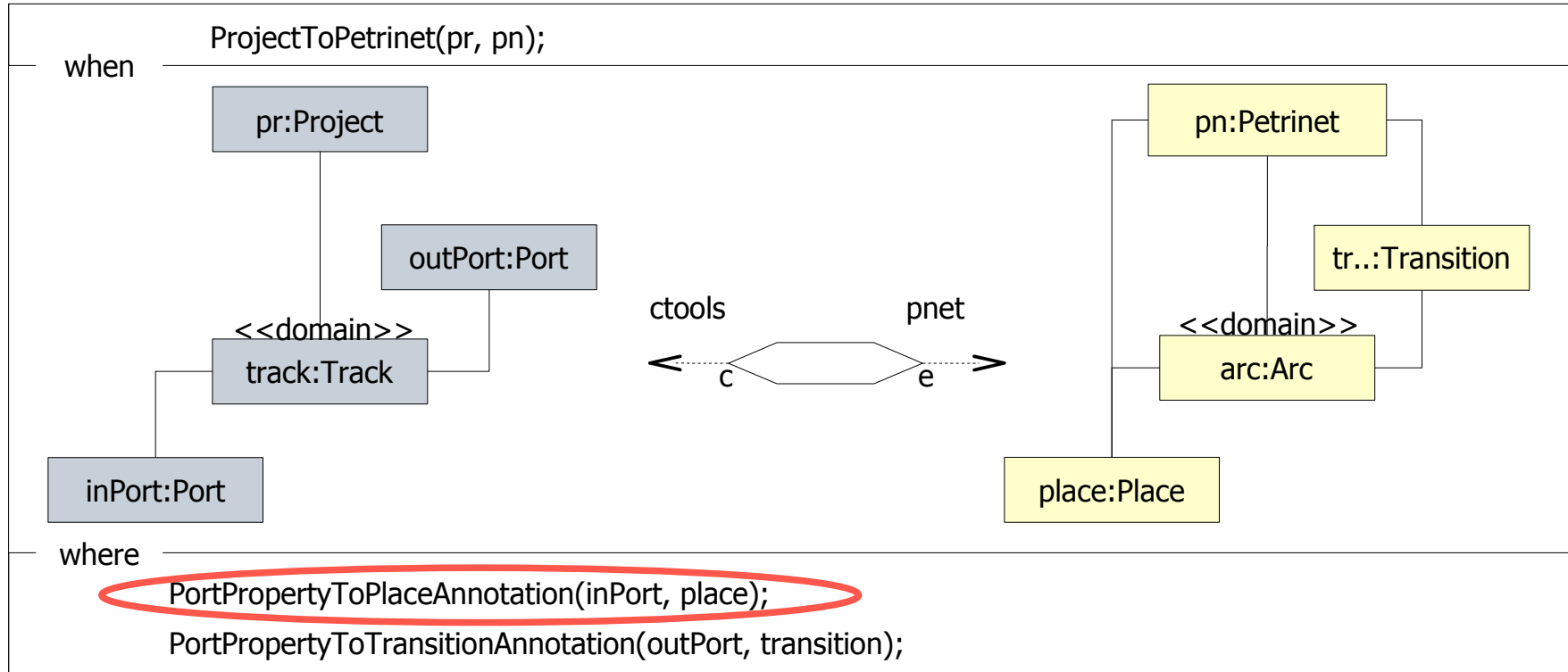
Different processing of TGGs,
but same result!
(Even more efficient...)

Mapping TrackToPlaceArcTransition



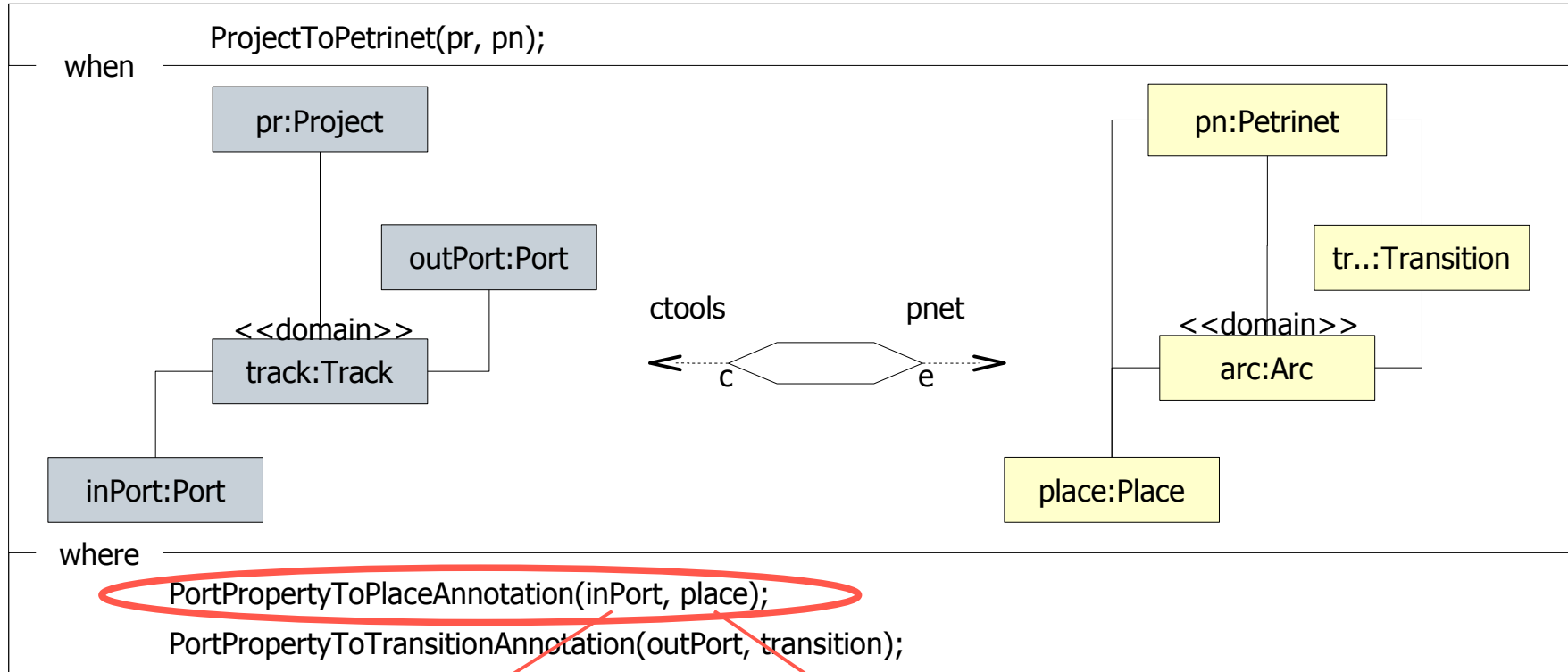
"where"-Relationships in QVT-Relations

Relation TrackToPlaceArcTransition

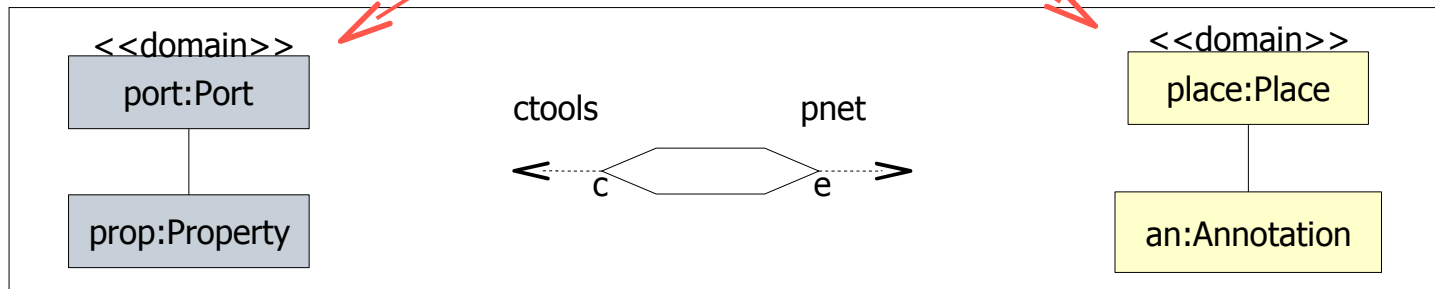


"where"-Relationships in QVT-Relations

Relation TrackToPlaceArcTransition

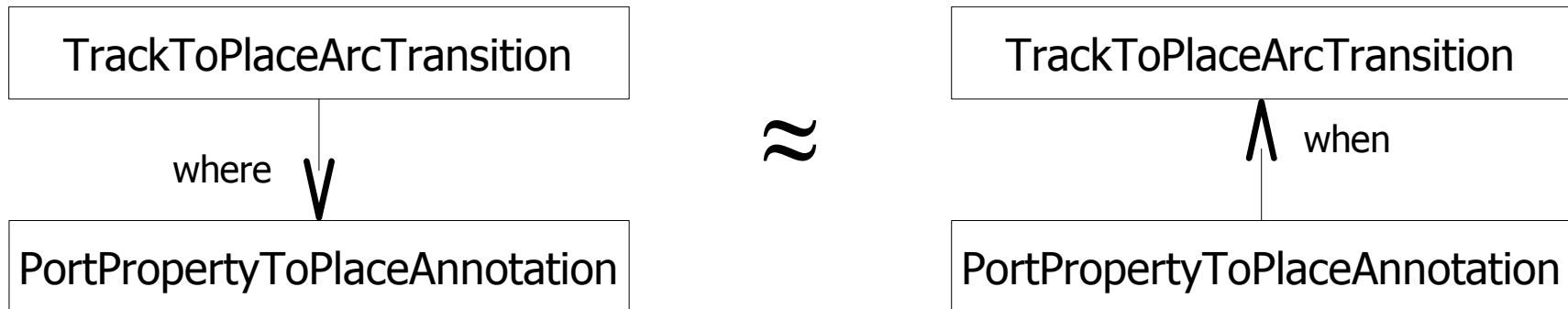


Relation TrackToPlaceArcTransition



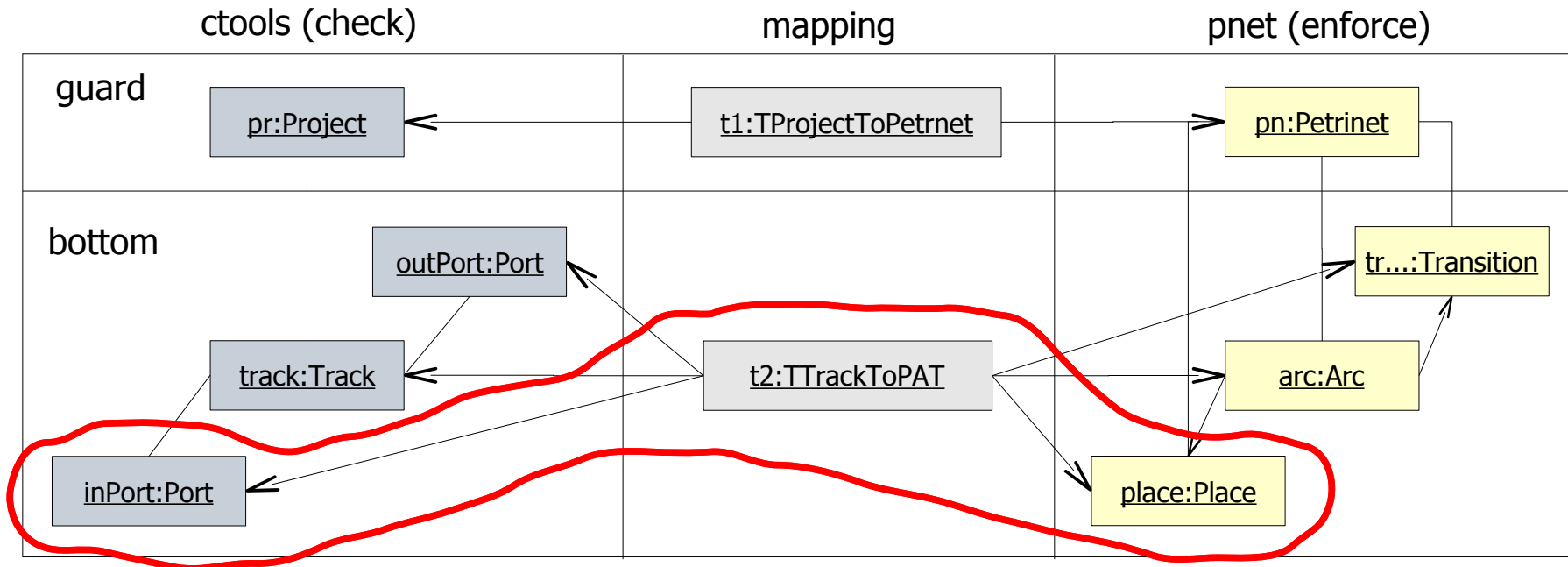
When vs. Where

- “Where” similar to a backward “When” of the invoked Relation:

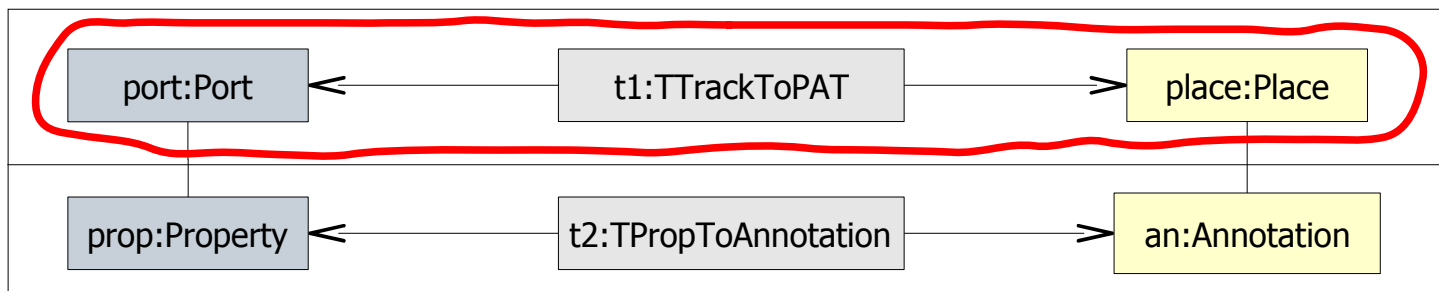


A "When"-Relationship in QVT-Core

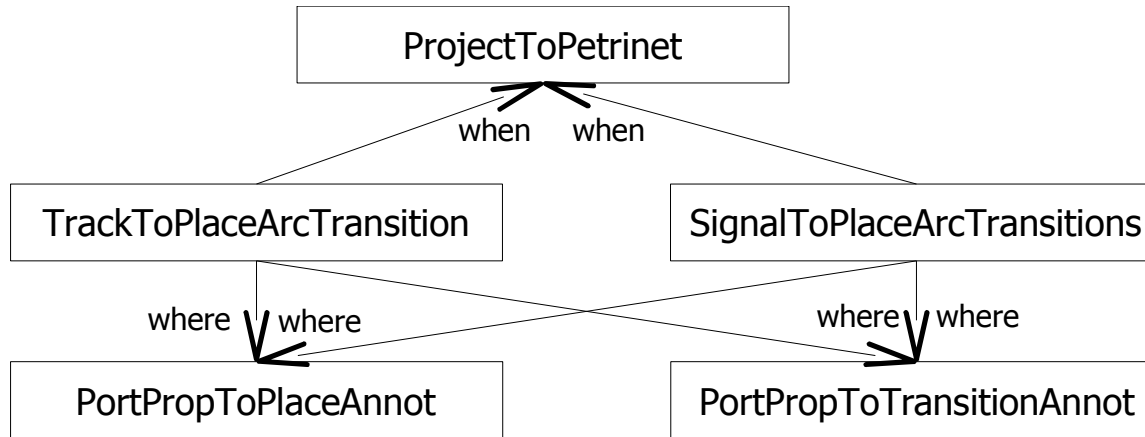
Mapping TrackToPlaceArcTransition



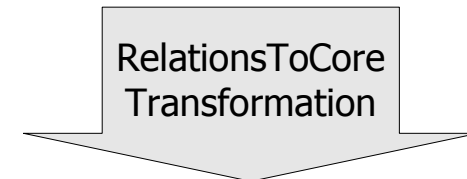
Mapping PortPropertyToPlaceAnnotation



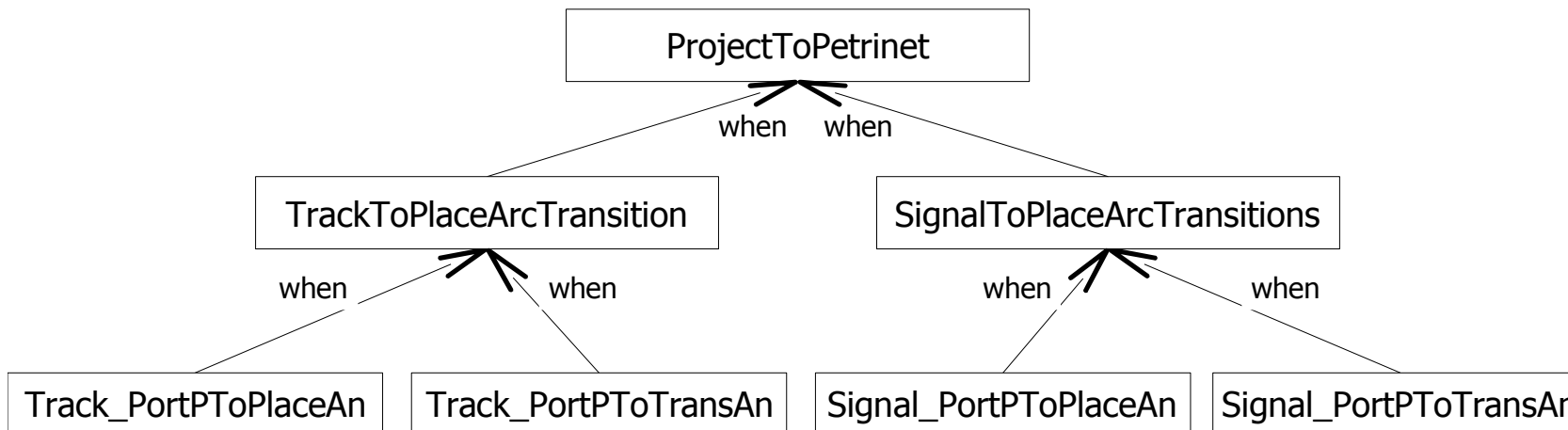
when&where Relationships in QVT-Core



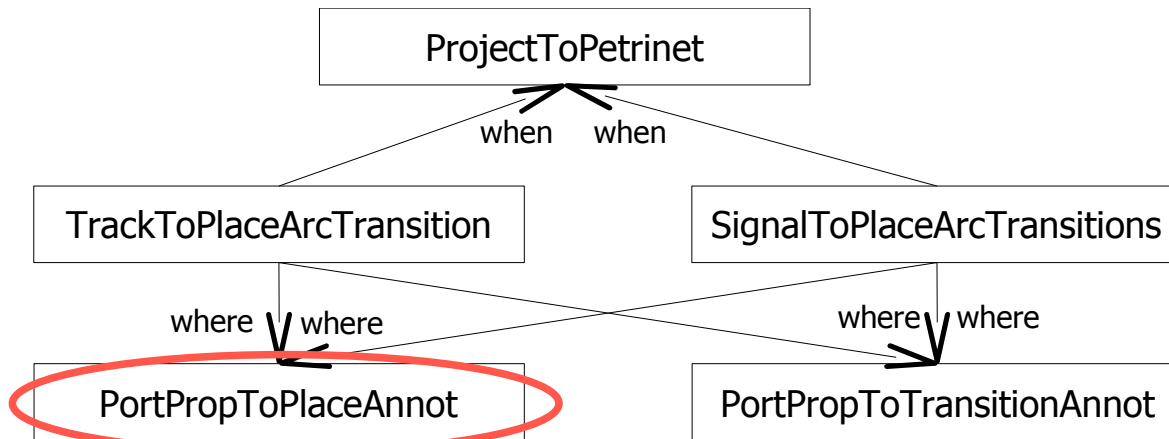
QVT-Relations



QVT-Core



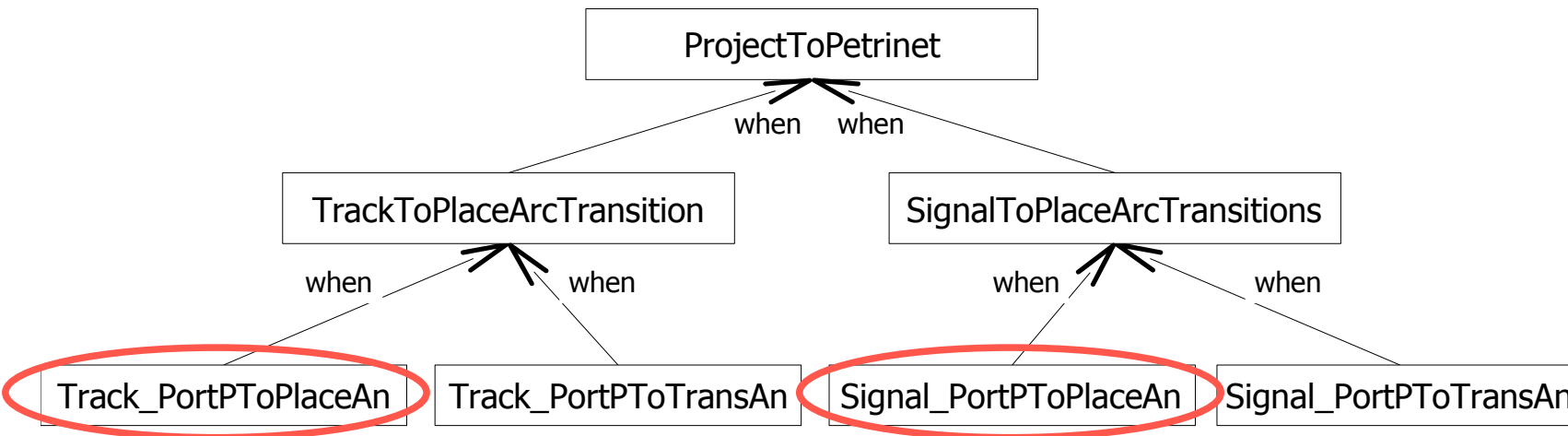
when&where Relationships in QVT-Core



QVT-Relations

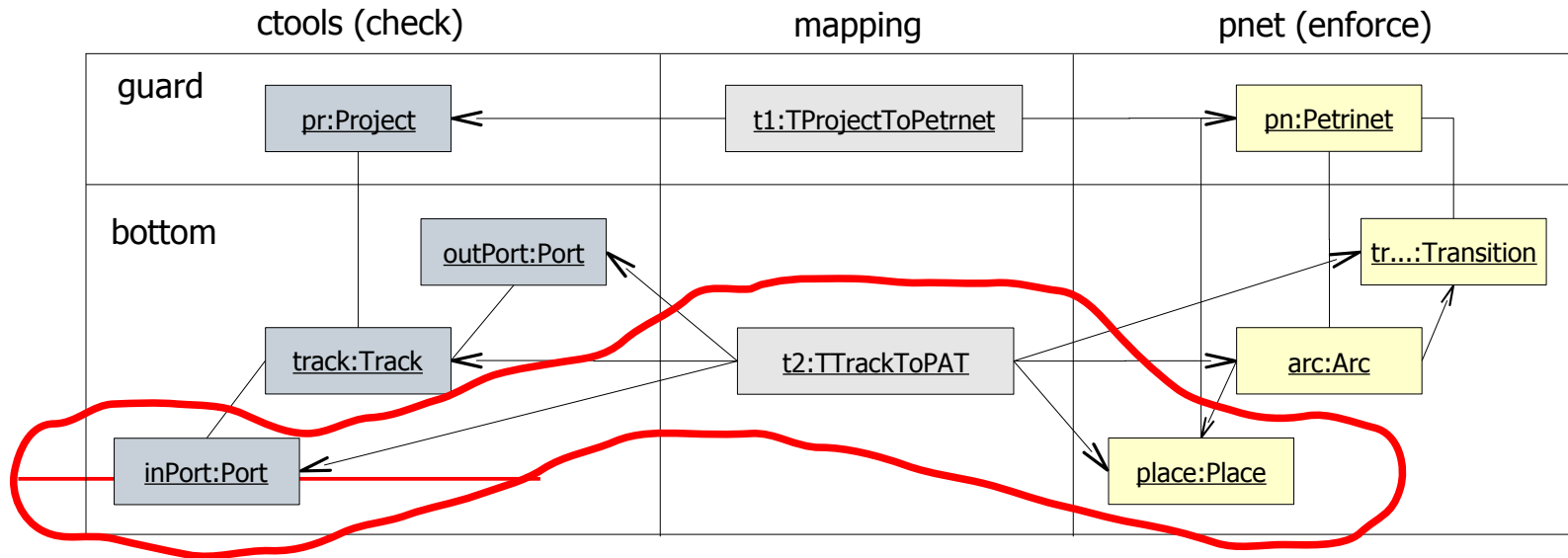
RelationsToCore Transformation

QVT-Core

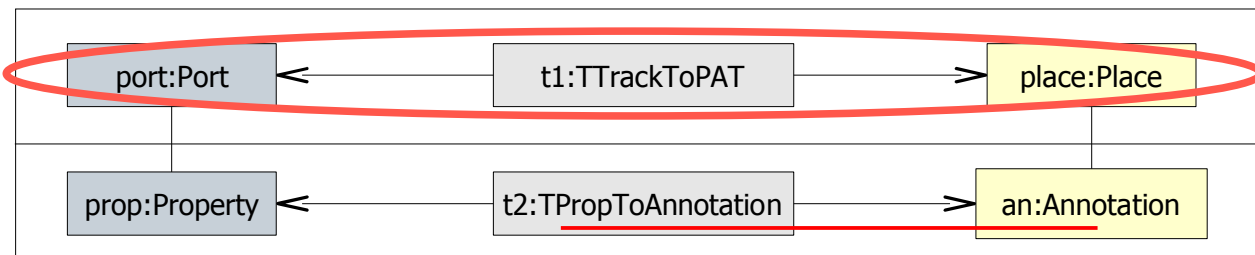


"Unfolded" Mappings

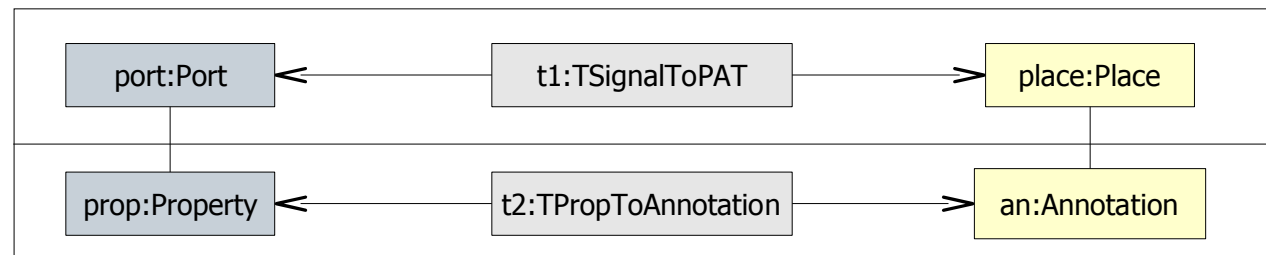
Mapping TrackToPlaceArcTransition



Mapping TrackToPlaceArcTransition_PortPropertyToPlaceAnnotation

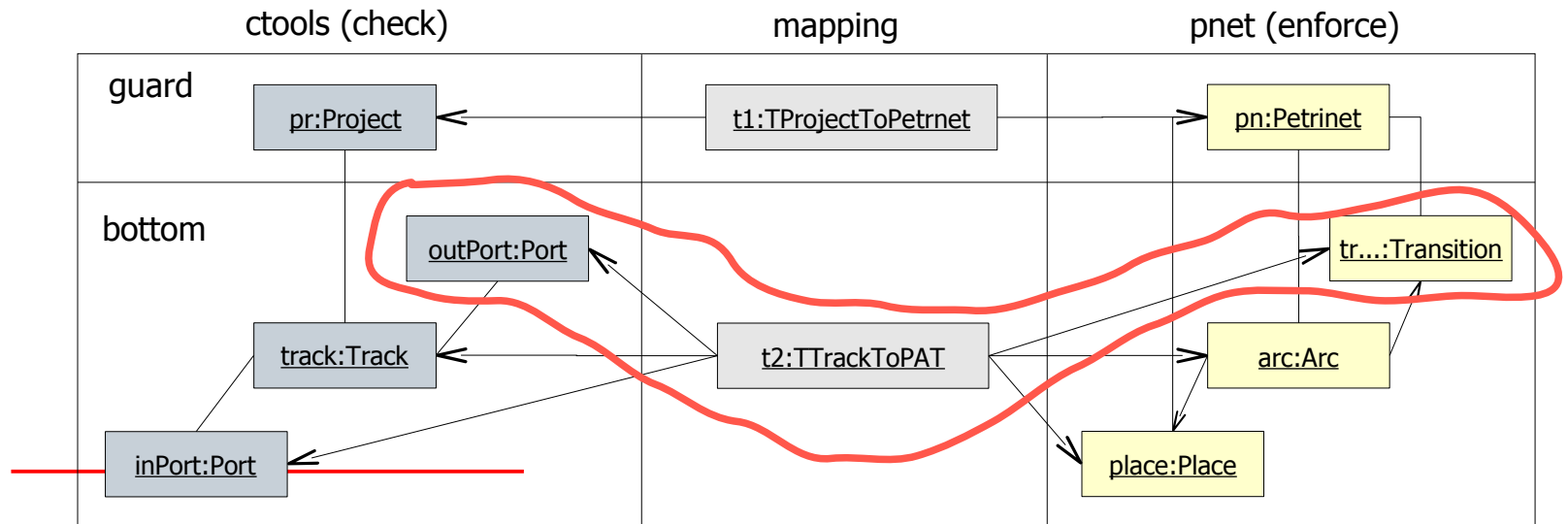


Mapping SignalToPlaceArcTransition_PortPropertyToPlaceAnnotation

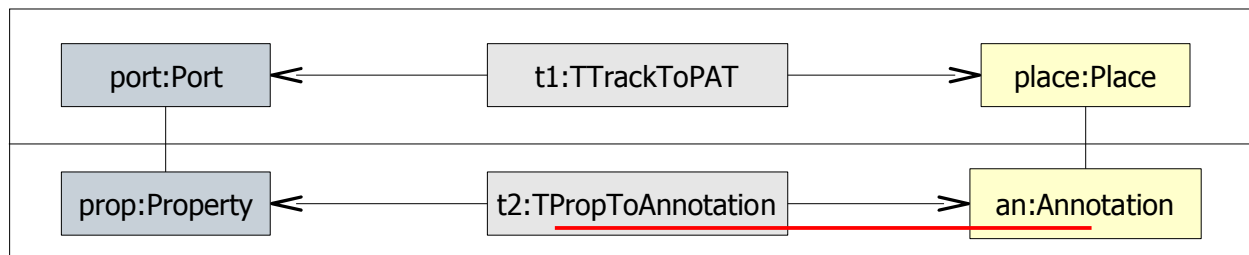


“Unfolded” Mappings

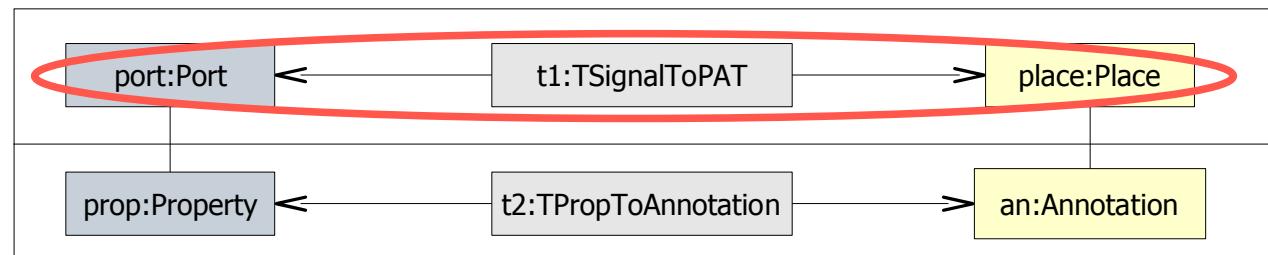
Mapping TrackToPlaceArcTransition



Mapping TrackToPlaceArcTransition_PortPropertyToPlaceAnnotation



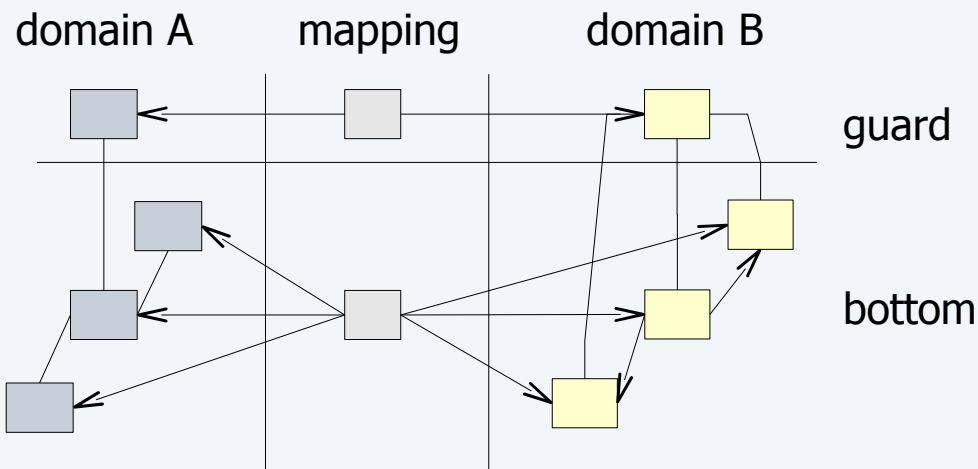
Mapping SignalToPlaceArcTransition_PortPropertyToPlaceAnnotation



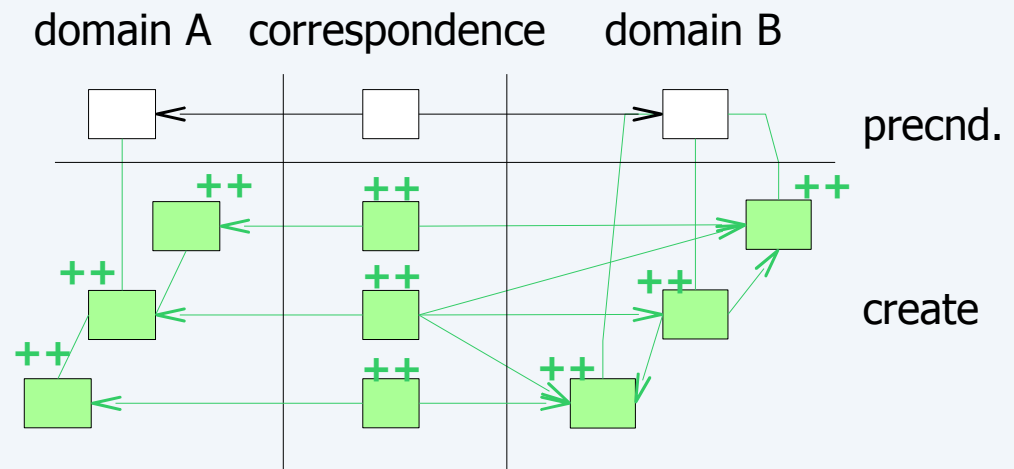
QVT-Core vs. TGGs – Short Comparison

- Structurally similar rules

QVT-Core*



TGG



- Graph structure: Variables and OCL-Predicates
- One *Trace Class*

- Graph structure: Nodes and Edges
- Multiple *Correspondence Nodes*

*:QVT-Core does not specify a graphical notation

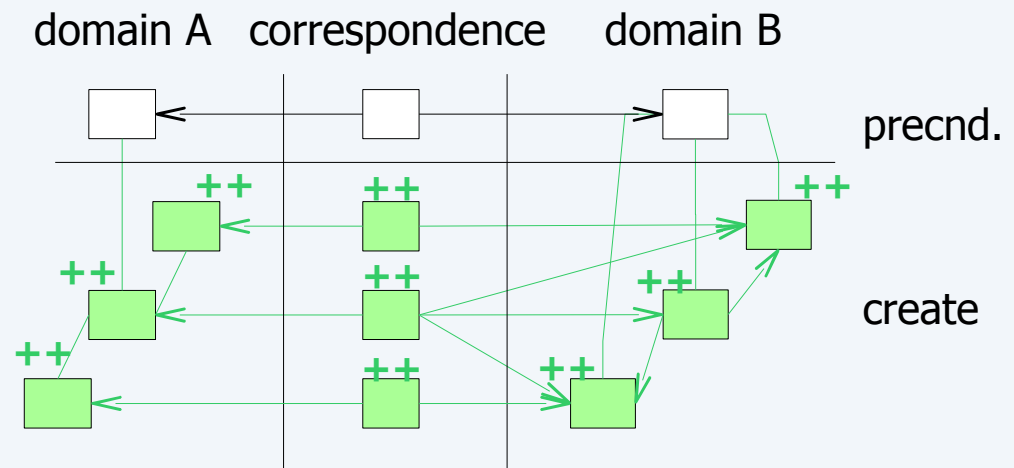
QVT-Core vs. TGGs – Short Comparison

- Structurally similar rules

QVT-Core

```
map TrackToPlaceArcTransition{  
  
  check ctools(project:Project){  
    track:Track, portIn:Port|  
    track.project = project;  
    track.port = portIn;  
    track.port = portOut;  
  }  
  
  check enforce pnet(petrinet:PetriNet){  
    realize place:Place, ...  
  }  
}
```

TGG



- Graph structure: Variables and OCL-Predicates
- One *Trace Class*

- Graph structure: Nodes and Edges
- Multiple *Correspondence Nodes*

QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

TGG

Node

QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

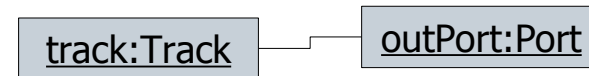
Reference Predicate (==) or
Reference Assignment (:=)

```
track.port:=outPort;
```

TGG

Node

Edge



QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

Reference Predicate (==) or
Reference Assignment (:=)

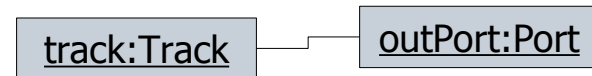
```
track.port := outPort;
```

(Simple) Attribute Value Predicate
or Attribute Value Assignment

TGG

Node

Edge



Attribute Value Constraint

QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

Reference Predicate (`==`) or
Reference Assignment (`:=`)

```
track.port := outPort;
```

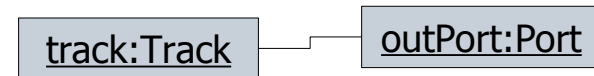
(Simple) Attribute Value Predicate
or Attribute Value Assignment

```
outPort.type == "out";
```

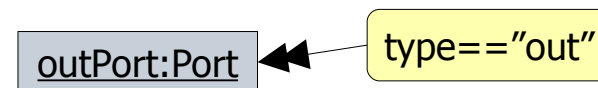
TGG

Node

Edge



Attribute Value Constraint



QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

Reference Predicate (==) or
Reference Assignment (:=)

```
track.port := outPort;
```

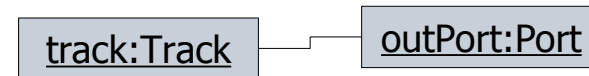
(Simple) Attribute Value Predicate
or Attribute Value Assignment

```
outPort.type == "out";
```

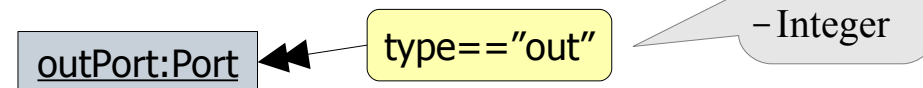
TGG

Node

Edge



Attribute Value Constraint



QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

Reference Predicate (==) or
Reference Assignment (:=)

```
track.port := outPort;
```

(Simple) Attribute Value Predicate
or Attribute Value Assignment

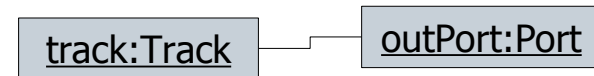
```
outPort.type == "out";
```

```
place.title := outPort.name;
```

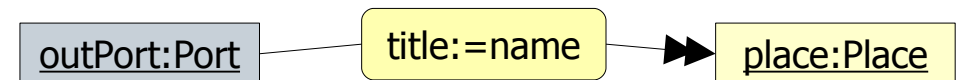
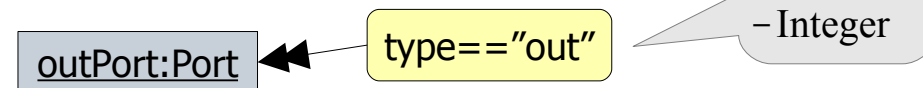
TGG

Node

Edge



Attribute Value Constraint



QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

Reference Predicate (==) or
Reference Assignment (:=)

```
track.port := outPort;
```

(Simple) Attribute Value Predicate
or Attribute Value Assignment

```
outPort.type == "out";
```

```
place.title := outPort.name;
```

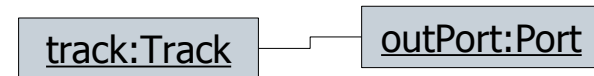
(Complex) Attribute Value
Predicate/Assignment

```
place.id := "id_" + outPort.name;
```

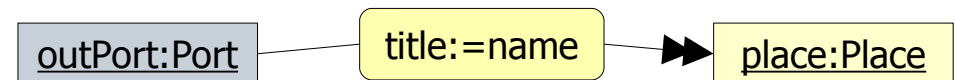
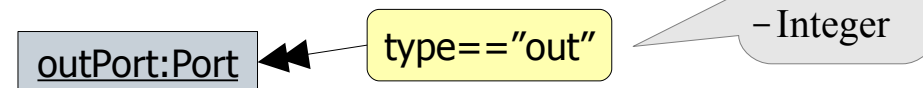
TGG

Node

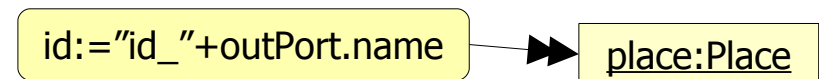
Edge



Attribute Value Constraint



OCL Attribute Value Constraint



QVT-Core vs. TGGs – Short Comparison

QVT-Core

Variable

Reference Predicate (==) or
Reference Assignment (:=)

```
track.port := outPort;
```

(Simple) Attribute Value Predicate
or Attribute Value Assignment

```
outPort.type == "out";
```

```
place.title := outPort.name;
```

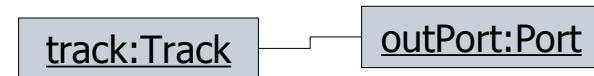
(Complex) Attribute Value
Predicate/Assignment

```
place.id := "id_" + outPort.name;
```

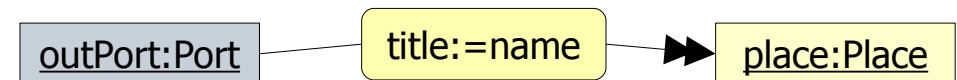
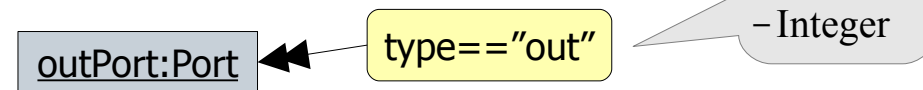
TGG

Node

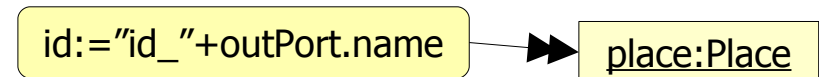
Edge



Attribute Value Constraint



Complex Attribute Value Constraint



Outlook

Summary

- QVT and TGGs are very similar
- TGGs still have advantages
 - Multiple Correspondence Nodes
 - Formal theory, e.g. to support verification
- Flexible, extendable Interpreter
 - Effectively used
 - For popular meta-modelling facility EMF (and also other Java models)
 - Flexible, e.g. n:m model transformations
- TGGs could be a valuable input for QVT

TGG Editor Screenshot

The screenshot displays the Eclipse IDE's TGG Editor. The main window shows a Petri net diagram with nodes for places, transitions, and arcs, connected by directed edges. The diagram is overlaid with a UML class diagram showing the underlying structure. Key classes include `project:Project`, `tProjectToPetriNet:TProjectToPetriNet`, `petriNet:PetriNet`, `track:Track`, `tTrackToPetriNet:TTrackToPetriNet`, `arc:Arc`, `place:Place`, `transition:Transition`, `inPort:Port`, and `outPort:Port`. The diagram also shows attribute constraints for `outPort:Port` and `inPort:Port`.

The Package Explorer on the left shows the project structure, including the `tracktopat.tgg_diagram` project. The Properties window at the bottom right shows the appearance and advanced properties of the selected element, such as the Slot Attribute, Slot Node, and Slot Value.

Property	Value
Slot Attribute	type : EString
Slot Attribute Name	type
Slot Node	Node outPort
Slot Value	Out
View	
Layout Constraint	
Styles	

TGG Editor – Creating a Node

1. Create a TGG node with the Node creation tool

2. Enter the node name

3. Draw an association to the DomainGraph-Pattern node

4. Select the desired type class from the domain class model

Property	Value
Represented Parameter	
Right	<input checked="" type="checkbox"/> true
Start Position	-1
Type	Track -> Component
Typed Name	Connection -> NamedElement
Type End Position	NamedElement
Type Start Position	Port -> NamedElement
	Project -> NamedElement
	Track -> Component

TGG Editor – Creating an Edge

The screenshot shows the TGG Editor interface with a diagram and a palette. The diagram features a source node 'pr:Project' and a target node 'track:Track'. A green arrow points from the source to the target, labeled with '++' and 'type unknown'. A blue box labeled '(Domain) ctools' is connected to both nodes. The 'Palette' on the right lists various elements, including 'Edge'. The 'Advanced' properties window at the bottom shows the configuration for the edge, with 'Type Reference' set to 'projectToComponent : Component'.

1. Draw an edge from a source to a target node

2. Select the appropriate type-reference

Property	Value
EMF	
Graph Pattern	Right Graph Pattern
Left	false
Name	
Right	true
Target Node	Node track
Typed Name	type_unknown
Type Reference	projectToComponent : Component
View	
Styles	

Starting a Transformation

correspondence model file

configuration file

new_configuration.interpreter

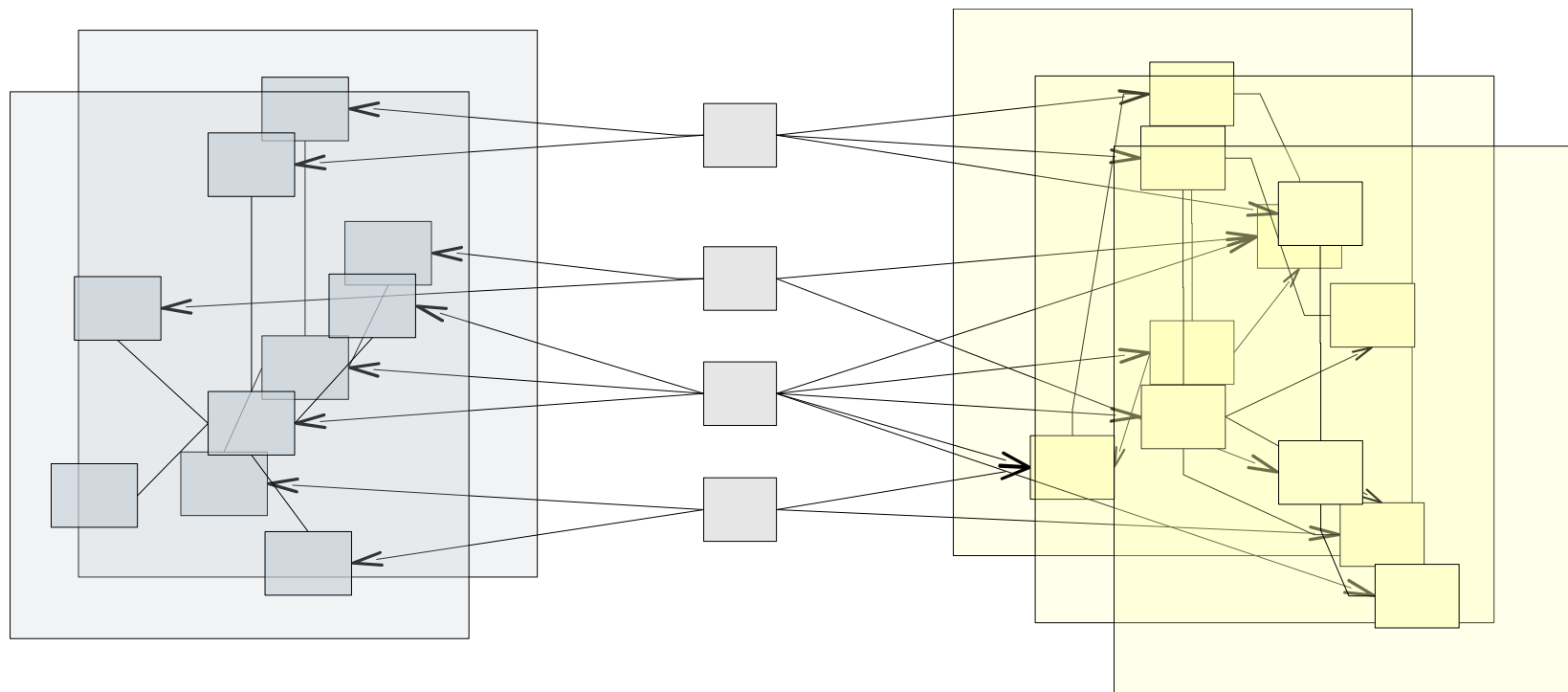
Start TGGInterpreter

CVS (ctools2pnetTransfc)

No changes in 'CVS (ctc)

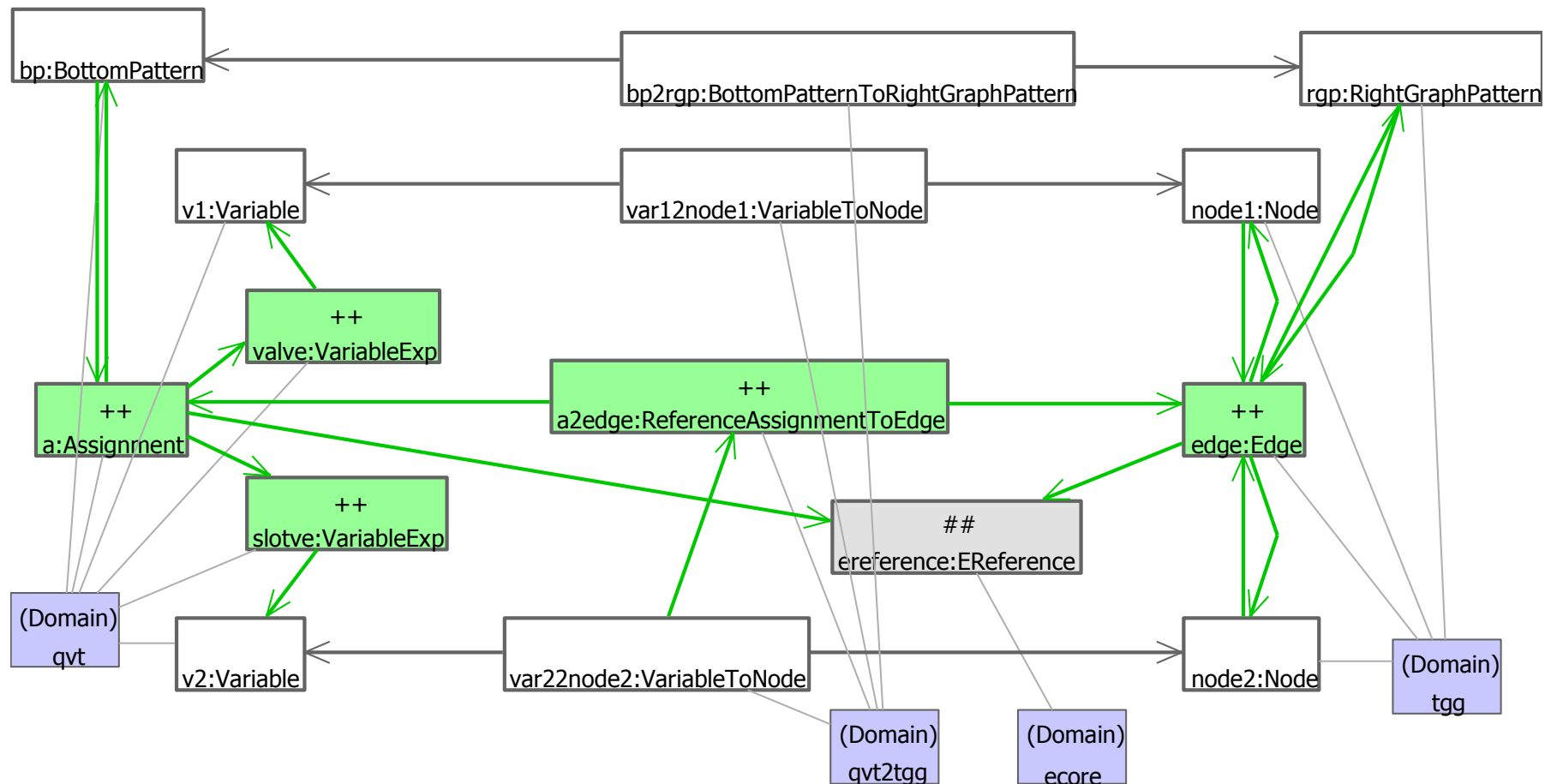
Further TGG-Interpreter Features

- n-to-m model transformations
- Multi Graph Grammars ("MGGs")



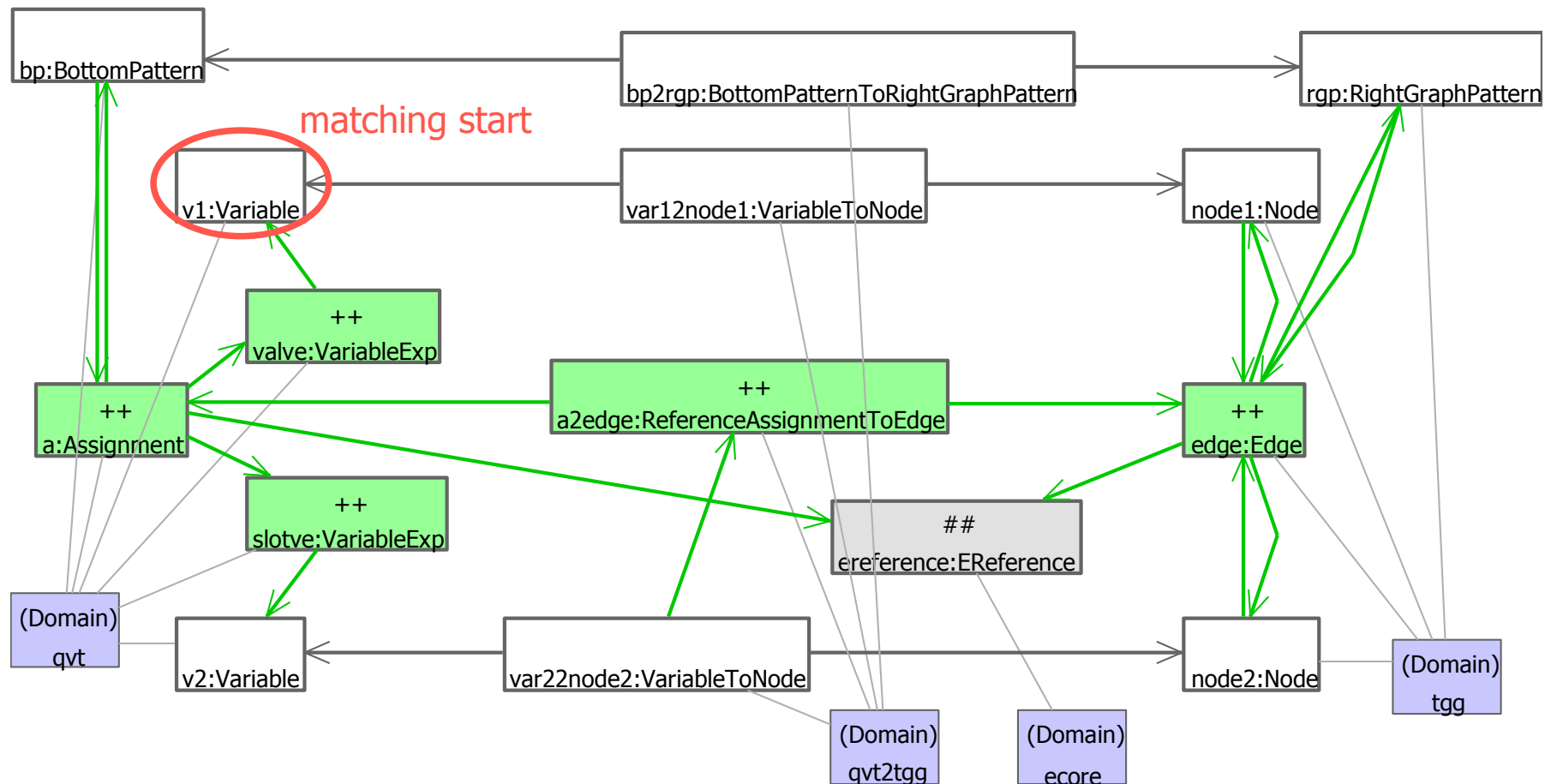
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



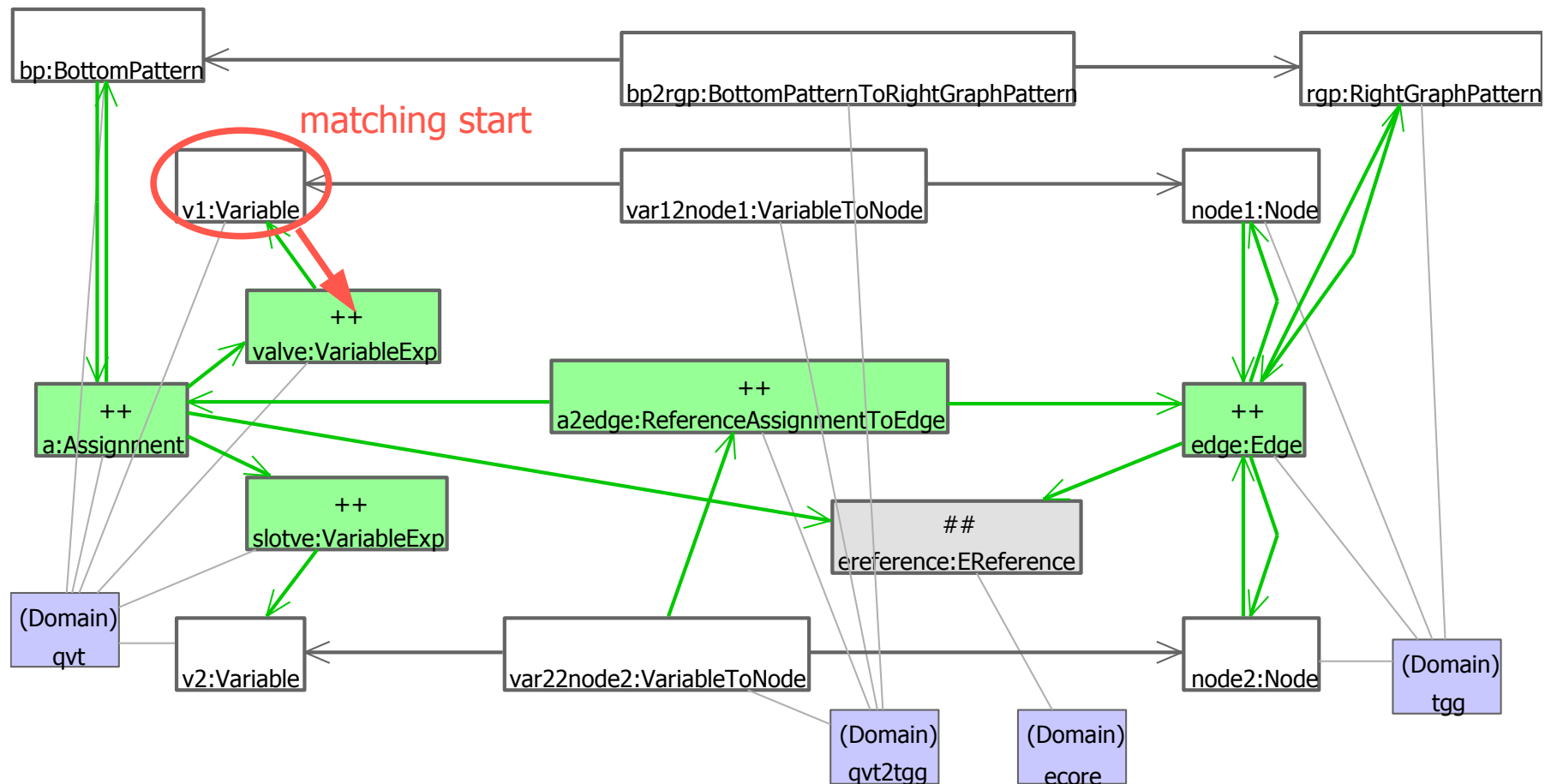
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



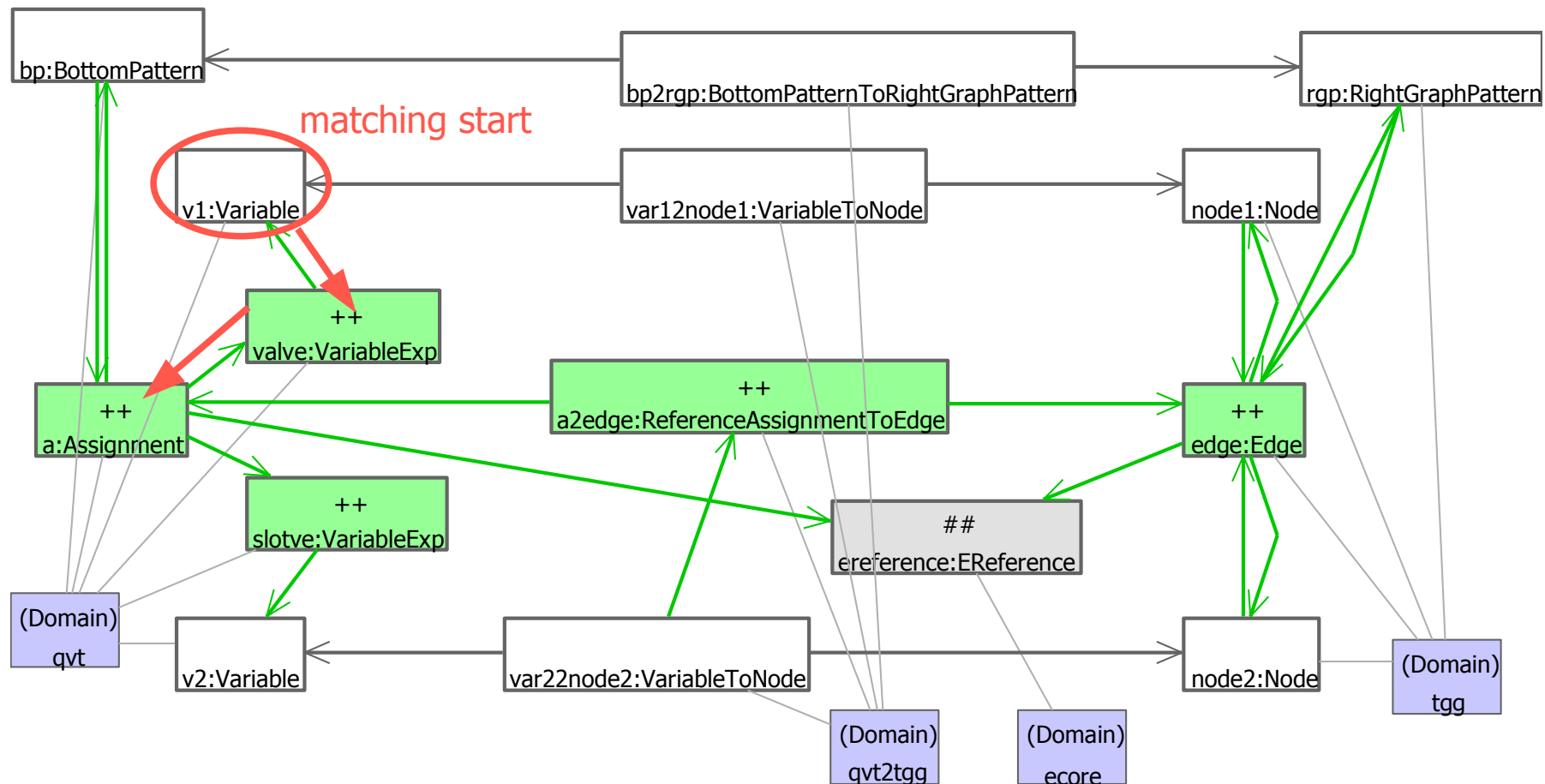
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



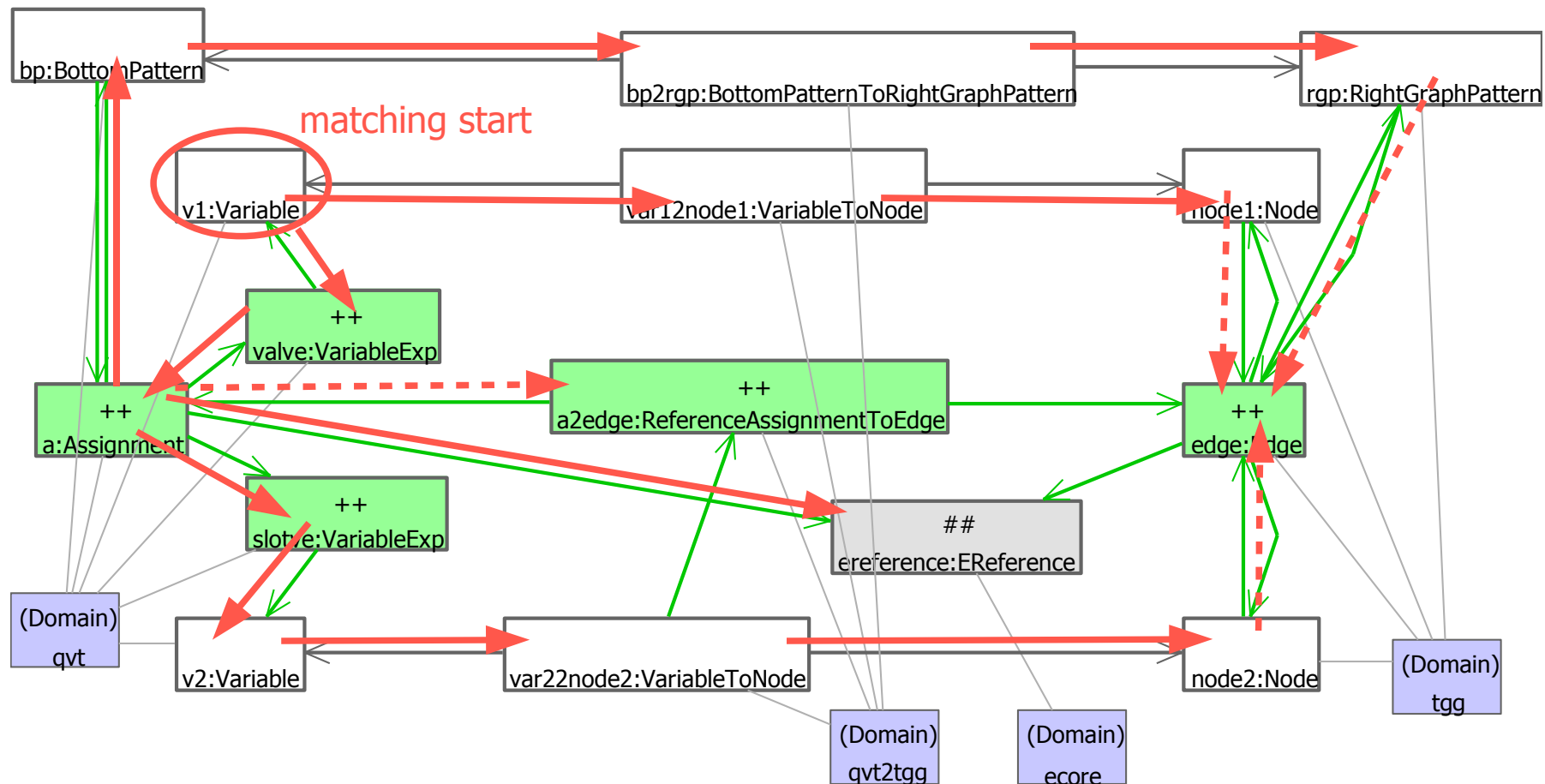
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



Bibliography

- [1] J. Greenyer: *A Study of Model Transformation Technologies: Reconciling TGGs with QVT*. University of Paderborn, July 2006. Master/Diploma thesis supervised by E. Kindler and R. Wagner
- [2] OMG: *MOF QVT Final Adopted Specification* <http://www.omg.org/docs/ptc/05-11-01.pdf>, 2005.
- [3] A. Schürr: Specification of Graph Translators with Triple Graph Grammars.. *Proceedings of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science*, Springer Verlag, Germany, June 1994.
- [4] H. Giese, R. Wagner: Incremental Model Synchronization with Triple Graph Grammars. *Proc. of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS)* Springer Verlag, (to appear), October 2006.
- [5] University of Paderborn, Germany: *Fujaba Tool Suite* <http://www.fujaba.de>, 2006.
- [6] A. Gepting, J. Greenyer, E. Kindler, A. Maas, S. Munkelt, C. Pales, T. Pivl, O. Rohe, V. Rubin, M. Sander, A. Scholand, C. Wagner, R. Wagner: Component Tools: A vision for a tool. *E. Kindler (ed.): Algorithmen und Werkzeuge für Petrinetze (AWPN) - Algorithms and Tools for Petri nets. Proceedings of the Workshop AWPN*, pp. 37-42, September 30th, October 1st, 2004.
- [7] Gepting A., Greenyer J., Maas A., Munkelt S., Pales C., Pivl T., Rohe O., Sanders M., Scholand A., Wagner C., *ComponentTools, an Infrastructure for the Design of Component Based Systems*, Informatiktage 2005, Congress of Computer Science, Sankt Augusting, Germany, April 2005.
- [8] A. Geburzi: *Synthese von Modelltransformationsregeln aus Uebersetzungsbeispielen (Synthesis of model transformation rules from examples)*. November, 2006. Master/Diploma thesis supervised by R. Wagner