2007 Bellairs
CAMPaM Workshop

A Study of Model Transformation Technologies:

Reconciling TGGs with QVT

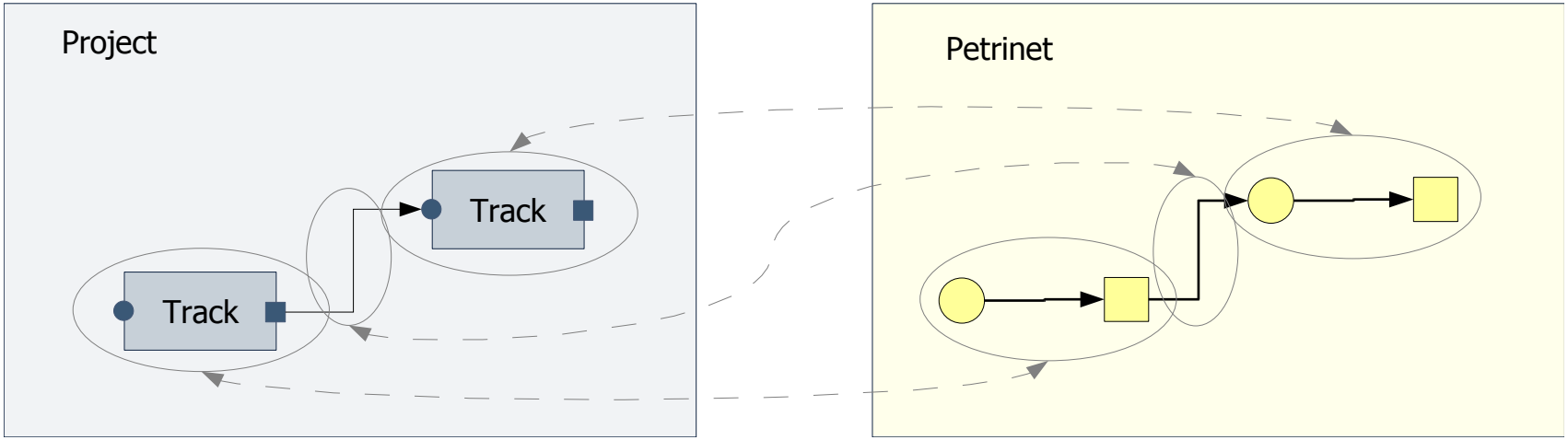
Joel Greenyer

Diploma Thesis supervised by Dr. Ekkart Kindler and Robert Wagner

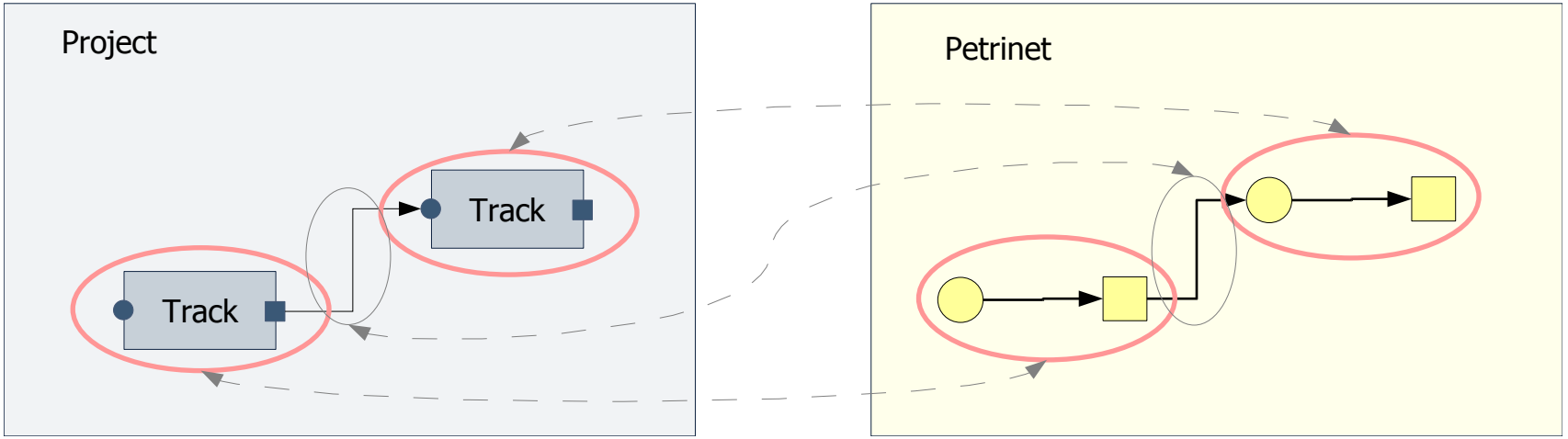
UNIVERSITÄT PADERBORN
Institut für Informatik – FG Softwaretechnik

Software Engineering Group

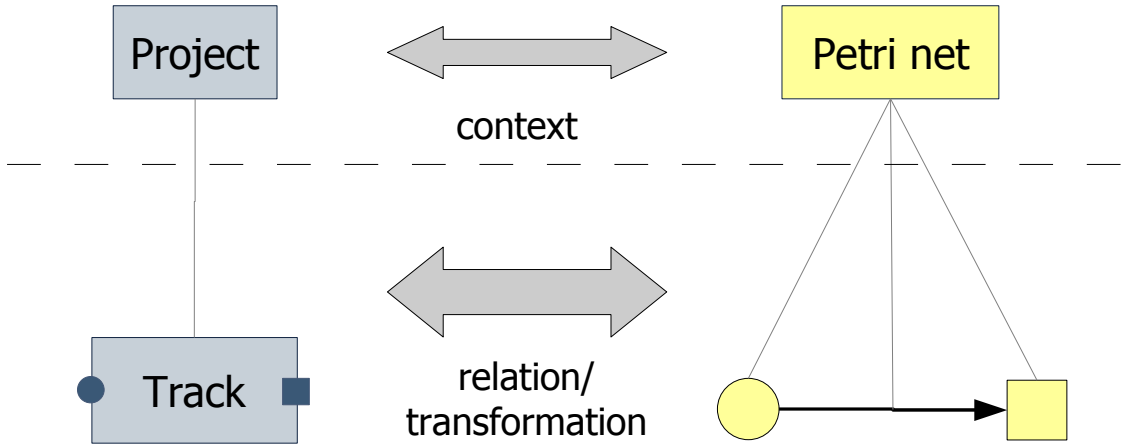
Example: ComponentTools To Petrinet



Example: ComponentTools To Petrinet

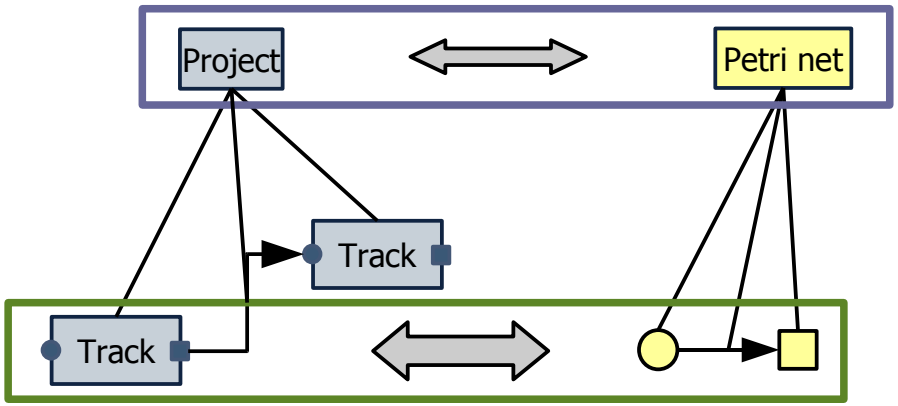


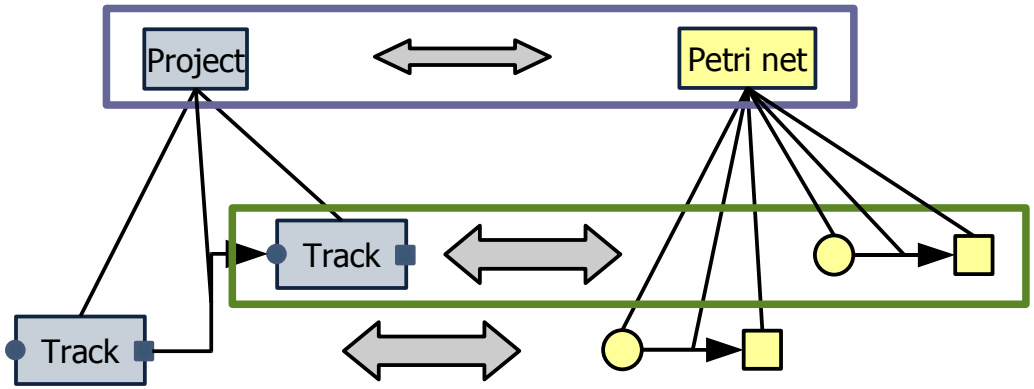
- Example Rule: TrackToPlaceArcTransition



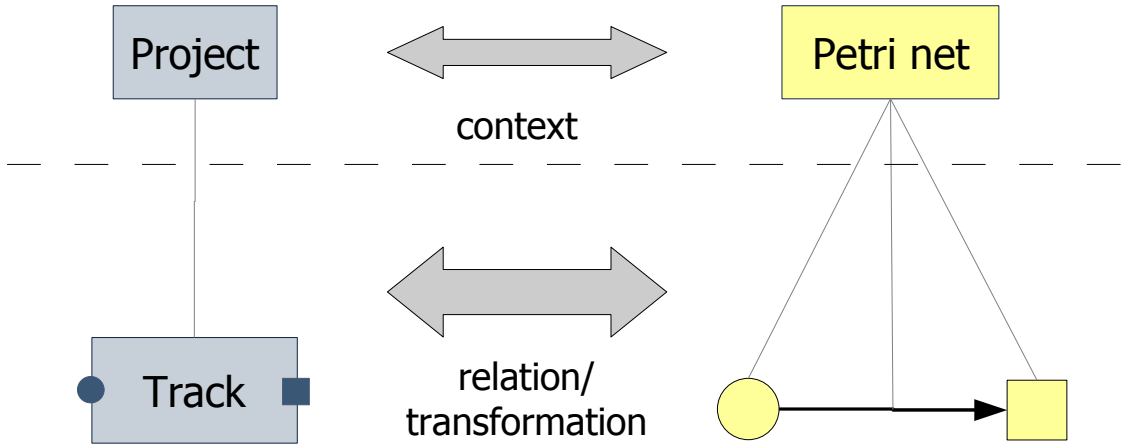
- Declarative
- Relational
- Graph-pattern-based

Example: ComponentTools To Petrinet





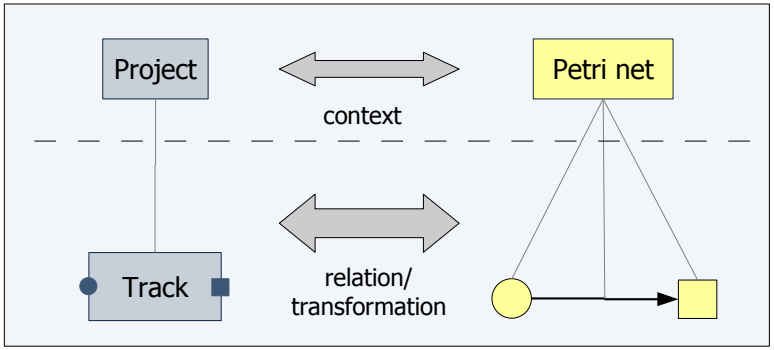
- Example Rule: TrackToPlaceArcTransition

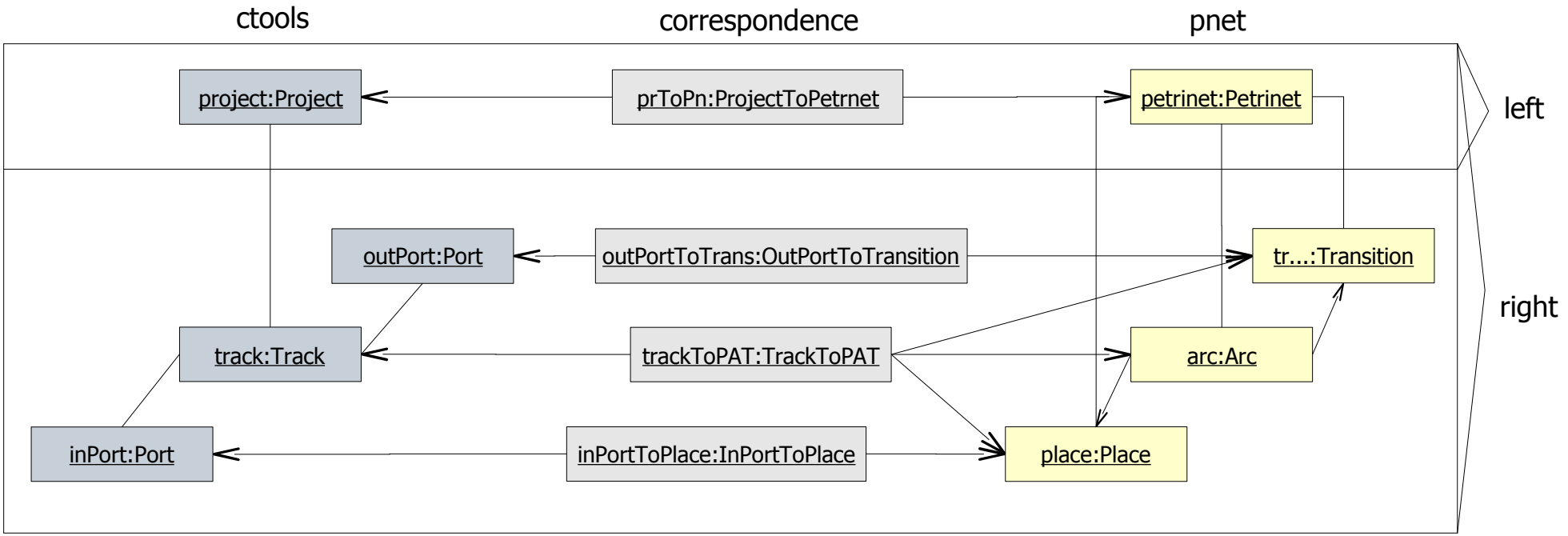


- Declarative
- Relational
- Graph-pattern-based

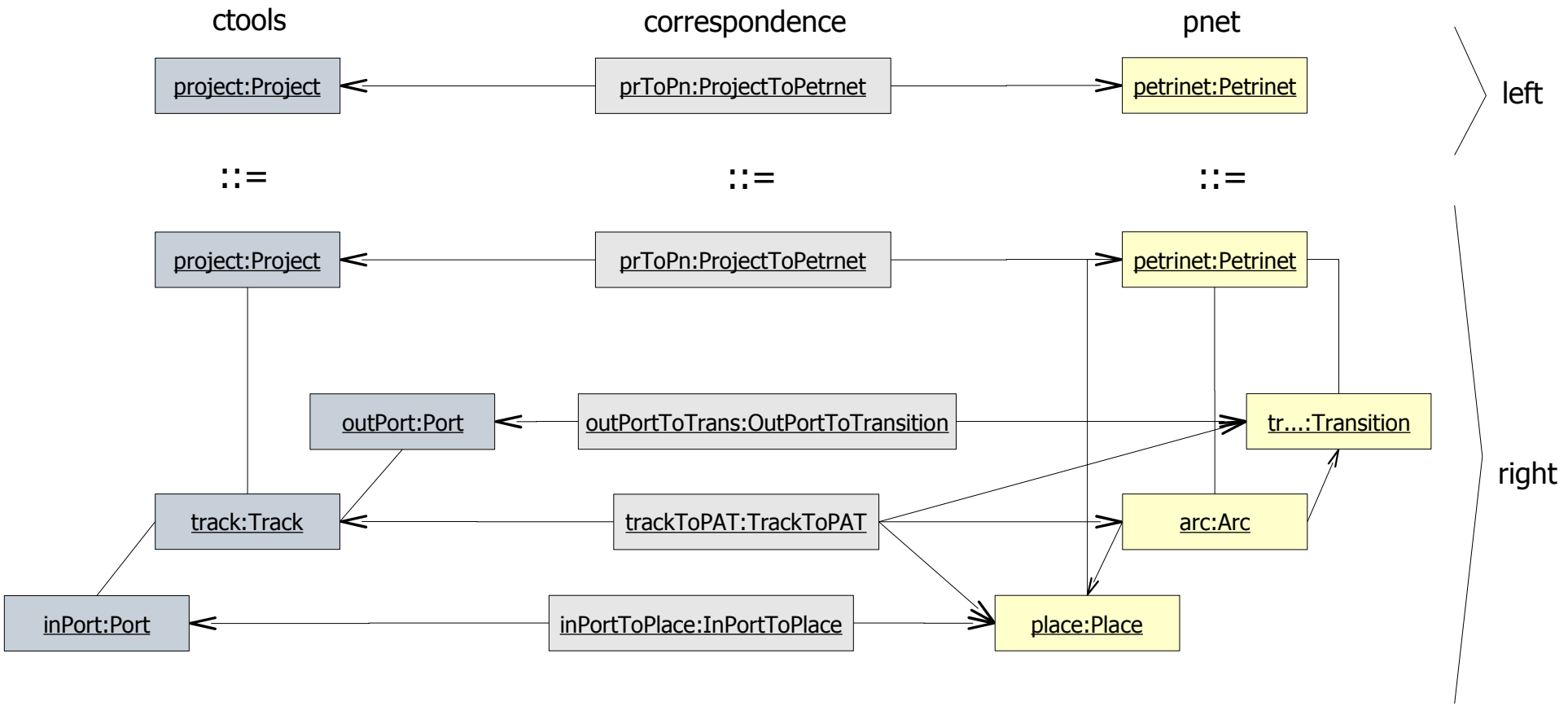
TGG Rule TrackToPlaceArcTransition

- The corresponding TGG rule:



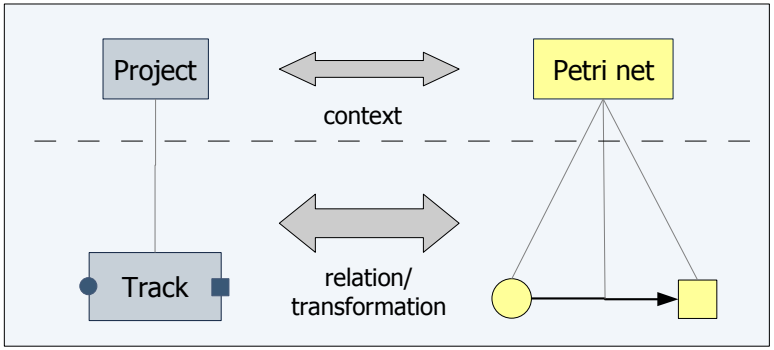


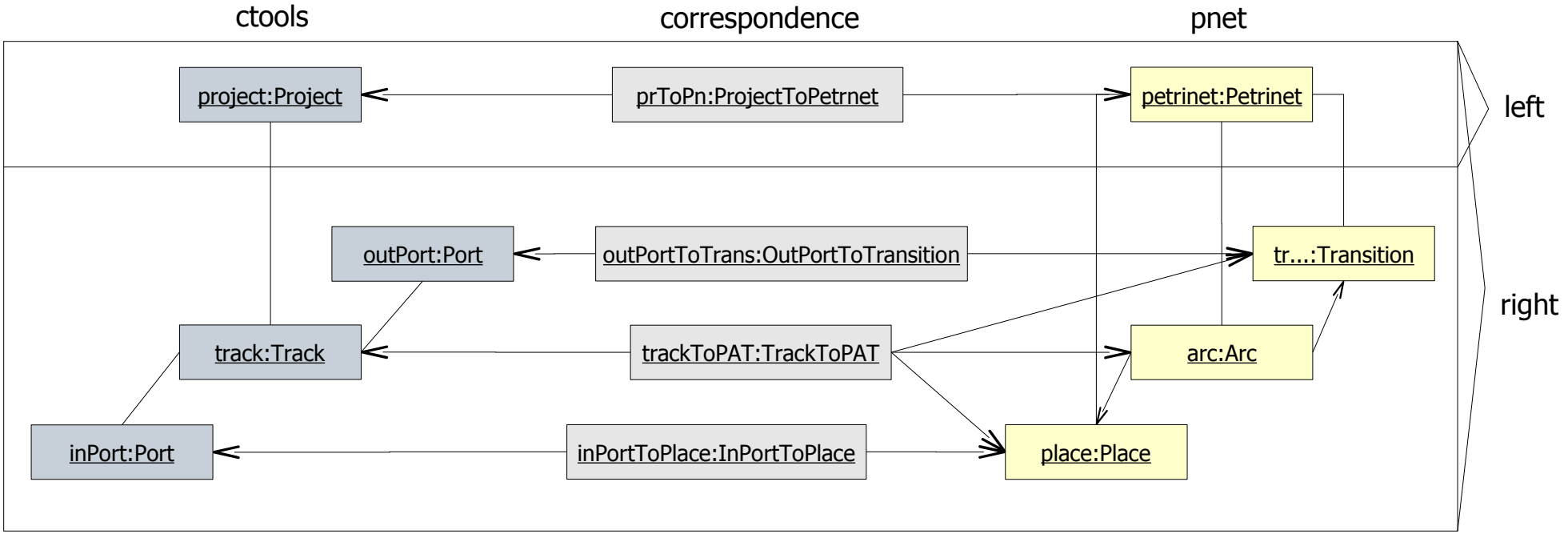
TGG Rule TrackToPlaceArcTransition



TGG Rule TrackToPlaceArcTransition

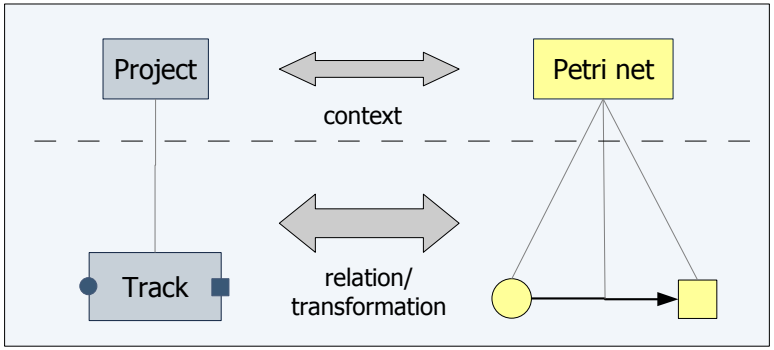
- The corresponding TGG rule:

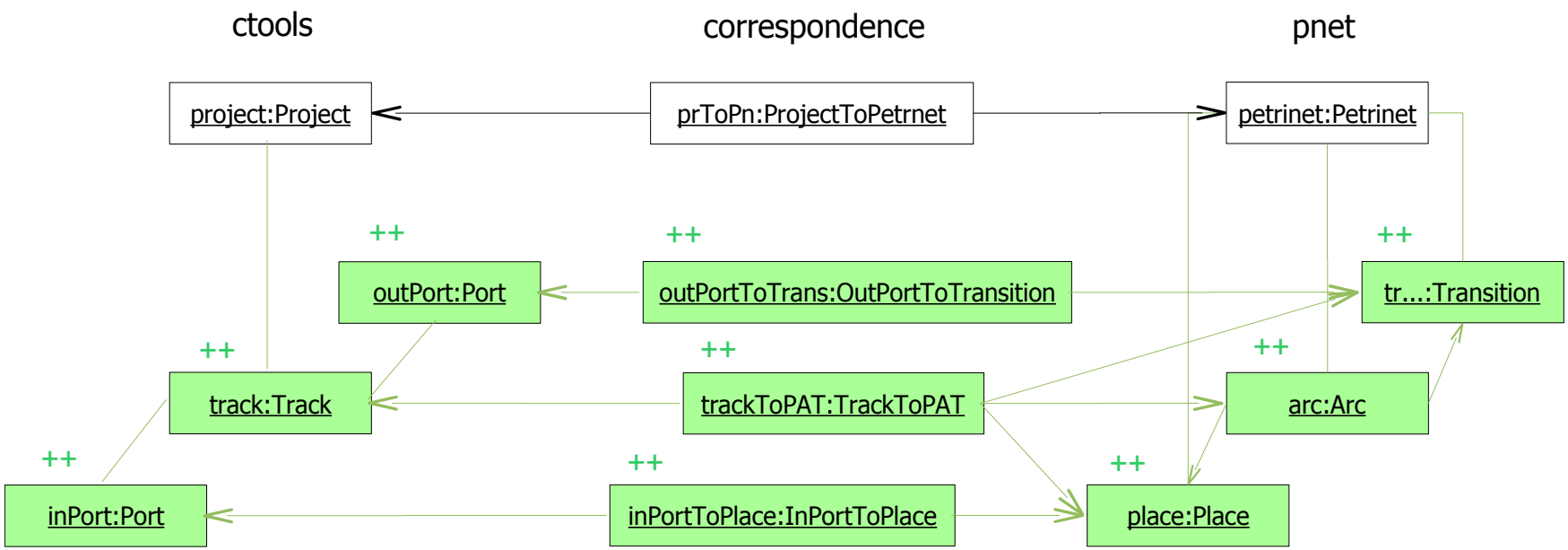




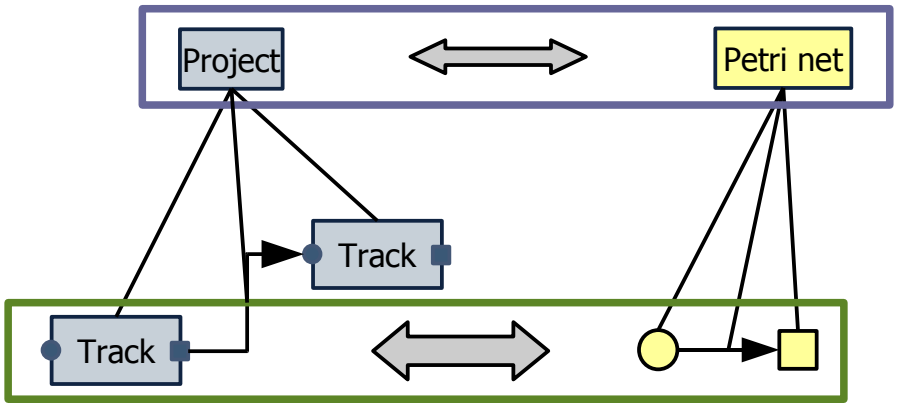
TGG Rule TrackToPlaceArcTransition

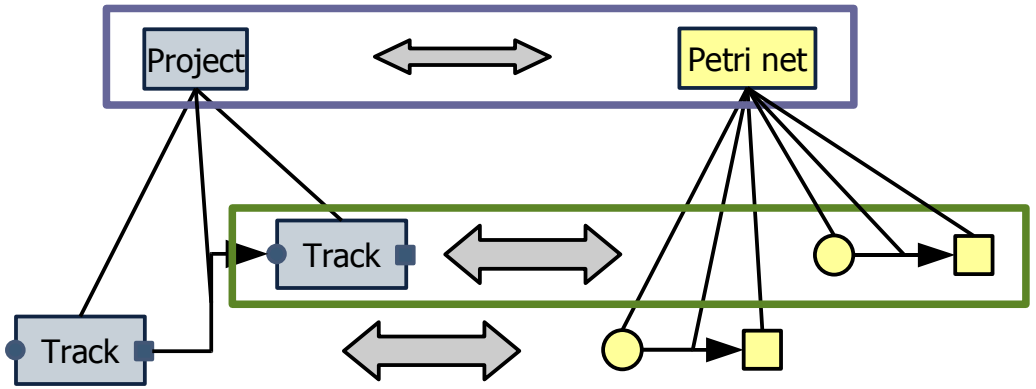
- The corresponding TGG rule:



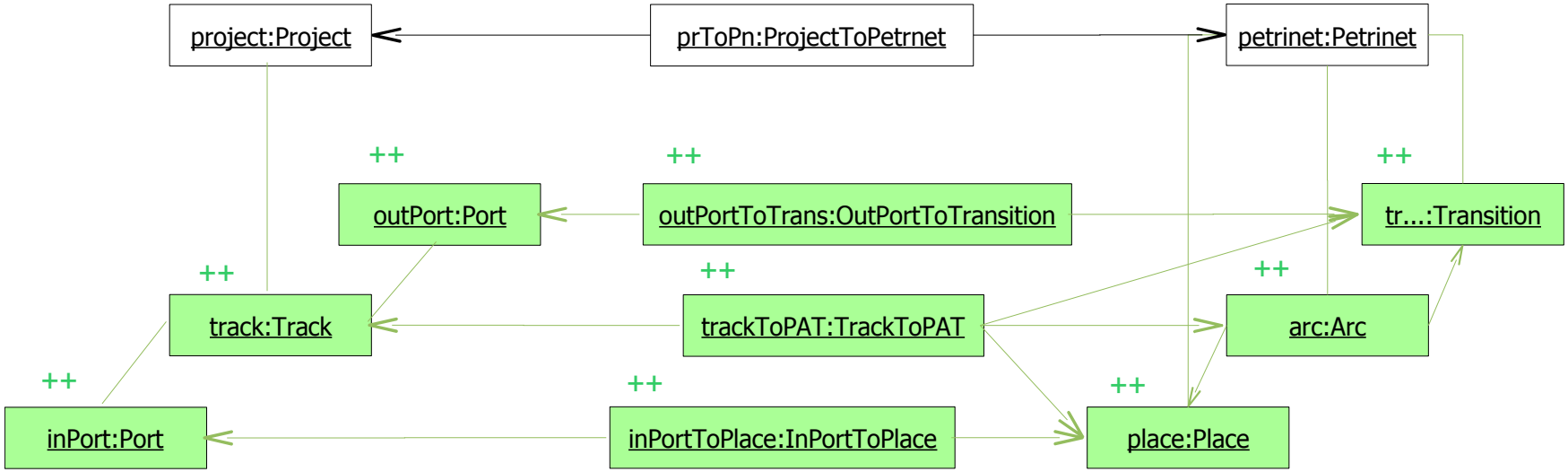


TGG Rule TrackToPlaceArcTransition

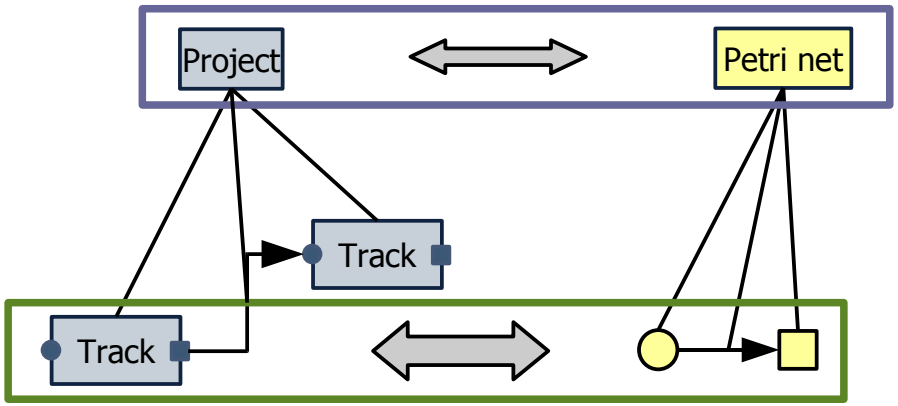


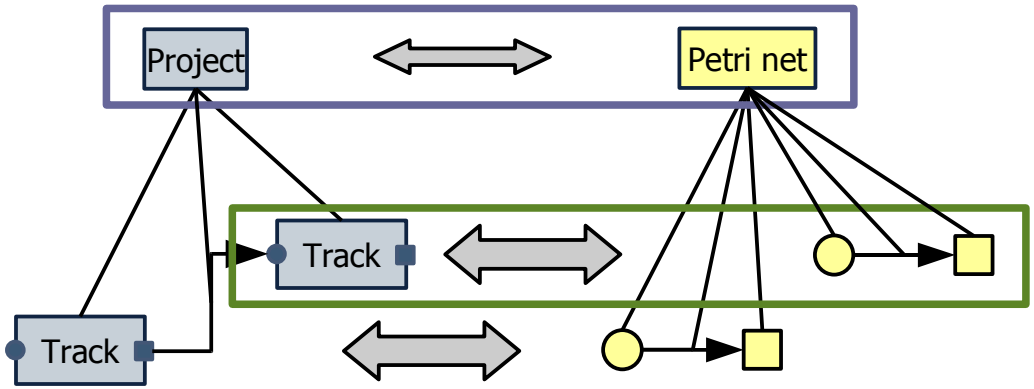


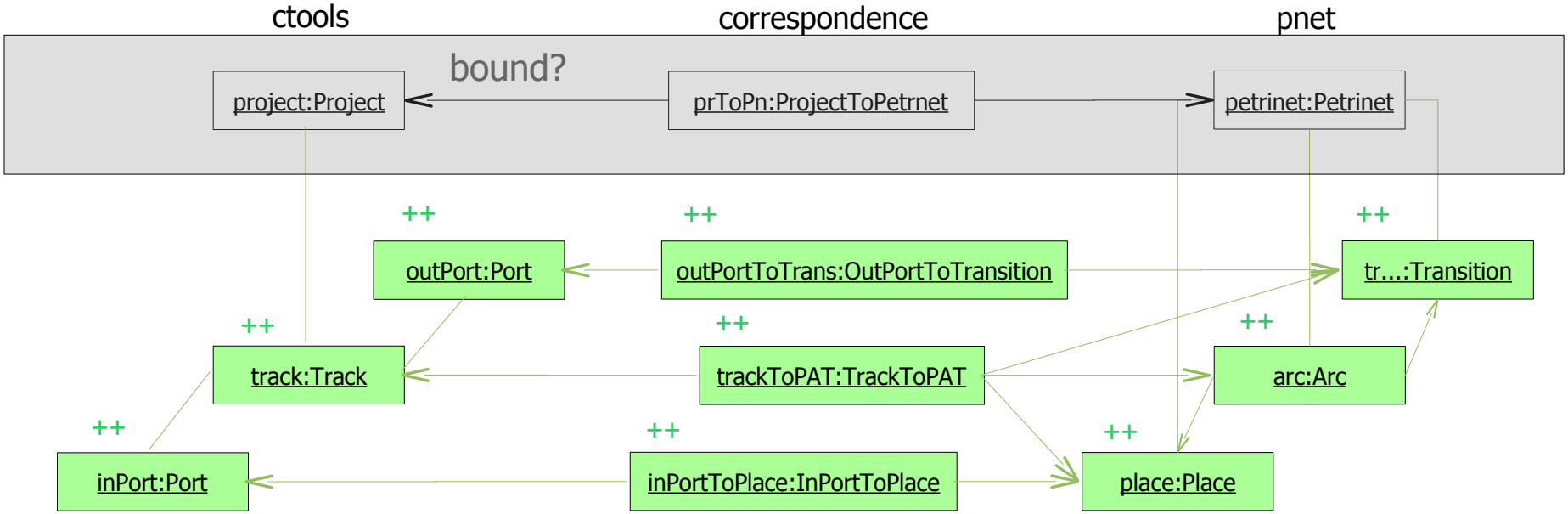
ctools correspondence pnet



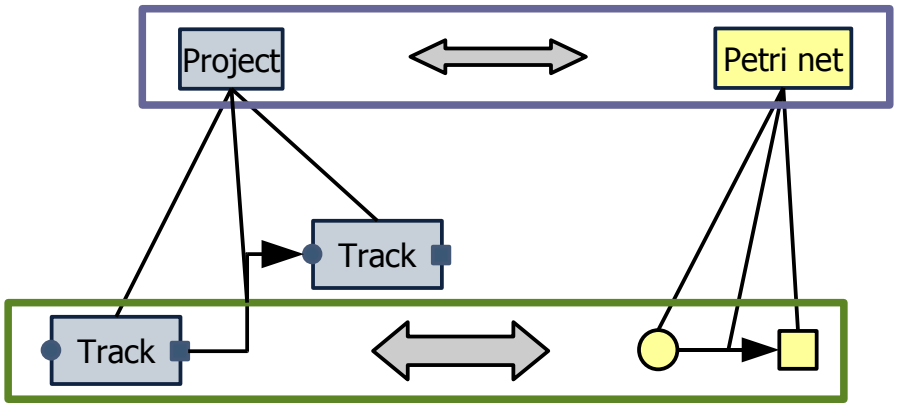
TGG Rule TrackToPlaceArcTransition

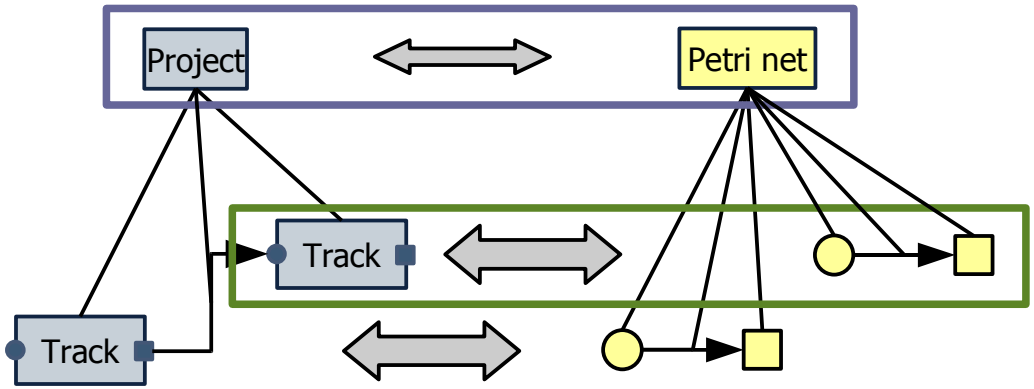


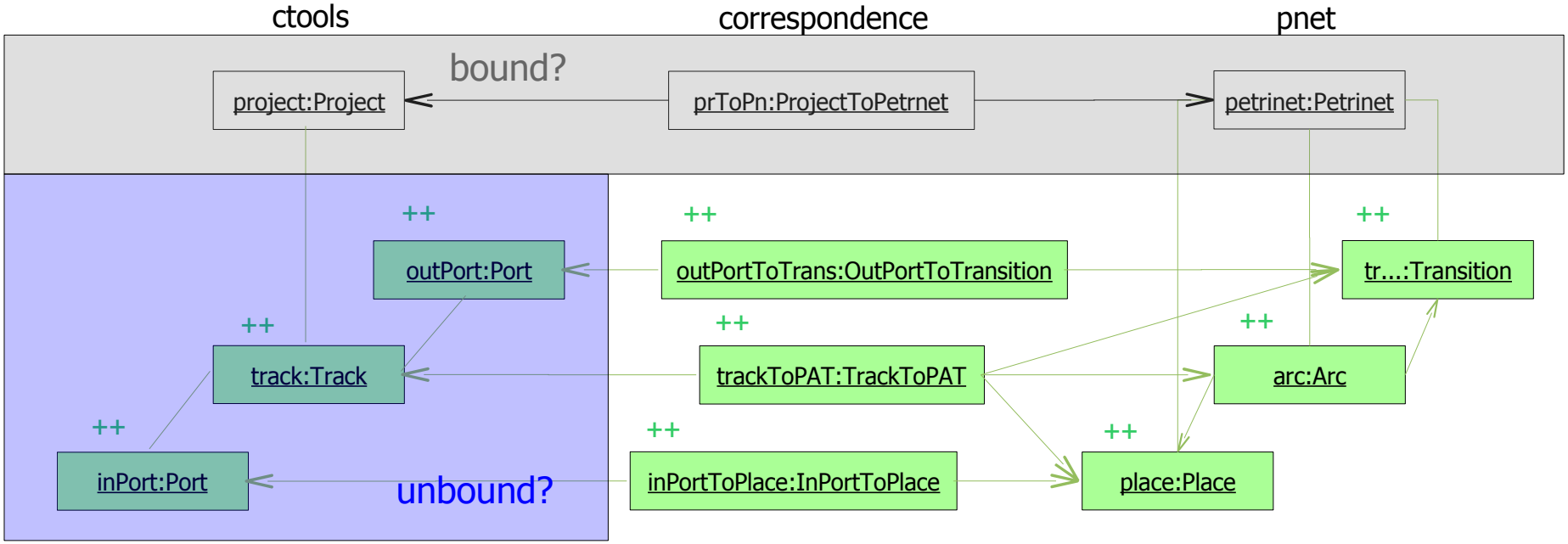




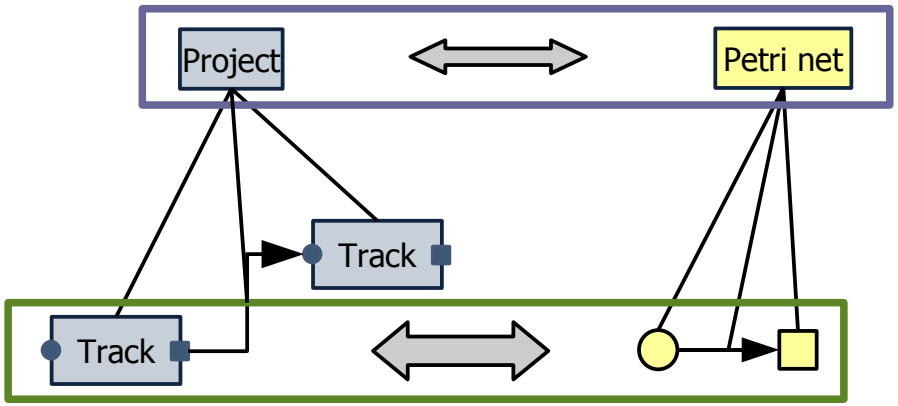
TGG Rule TrackToPlaceArcTransition

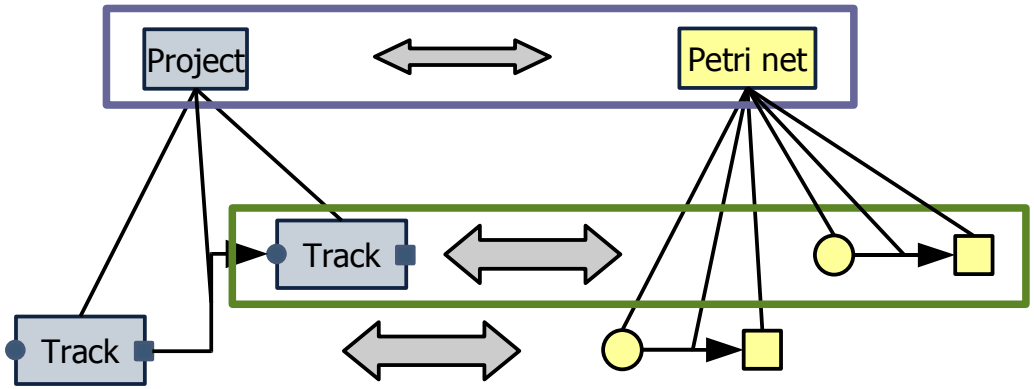


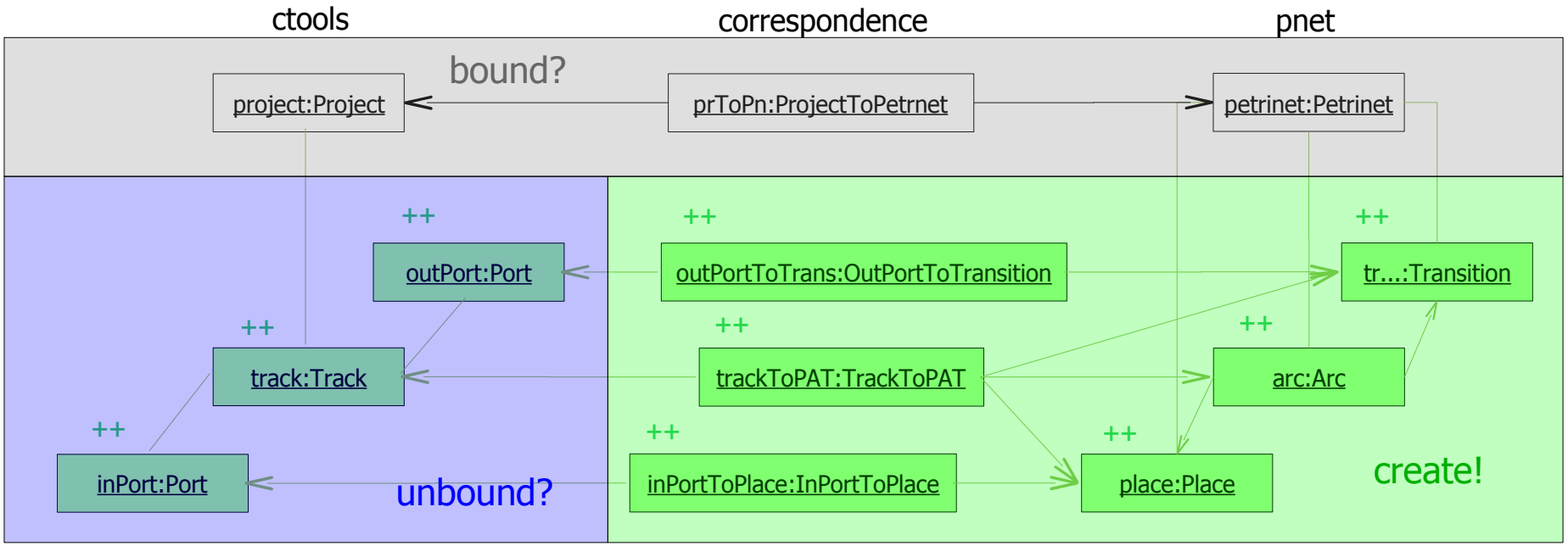




TGG Rule TrackToPlaceArcTransition

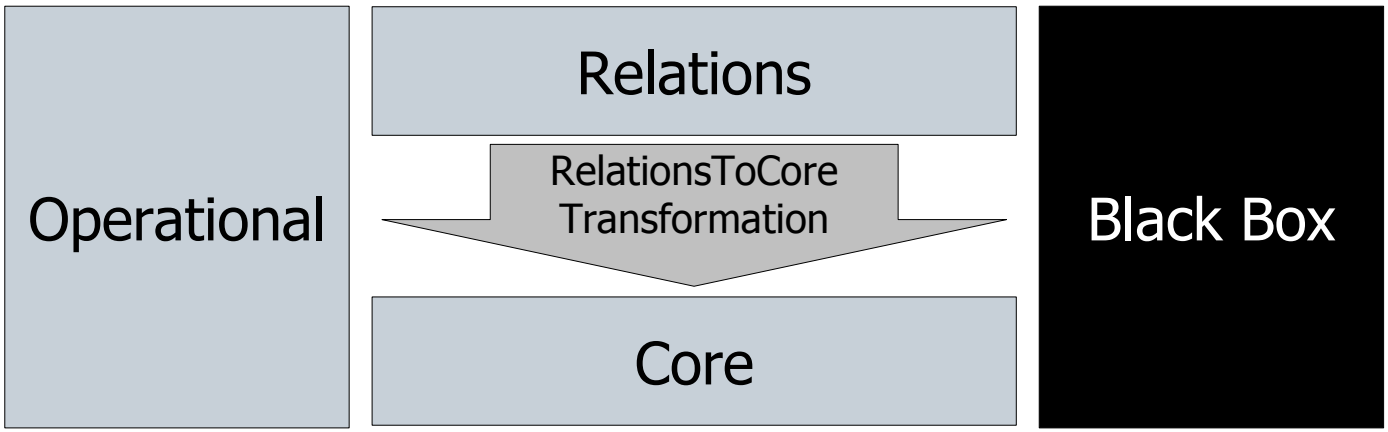






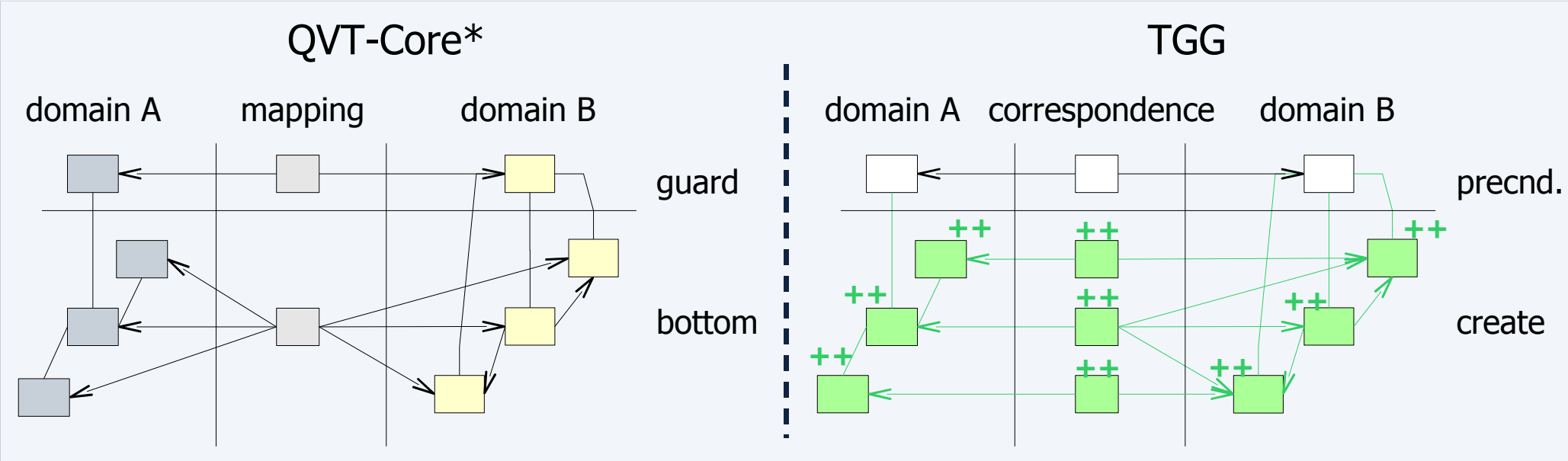
QVT Overview

- Model Transformation Approaches in MDD and MDA
  - Many techniques, different strenghts 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



- 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

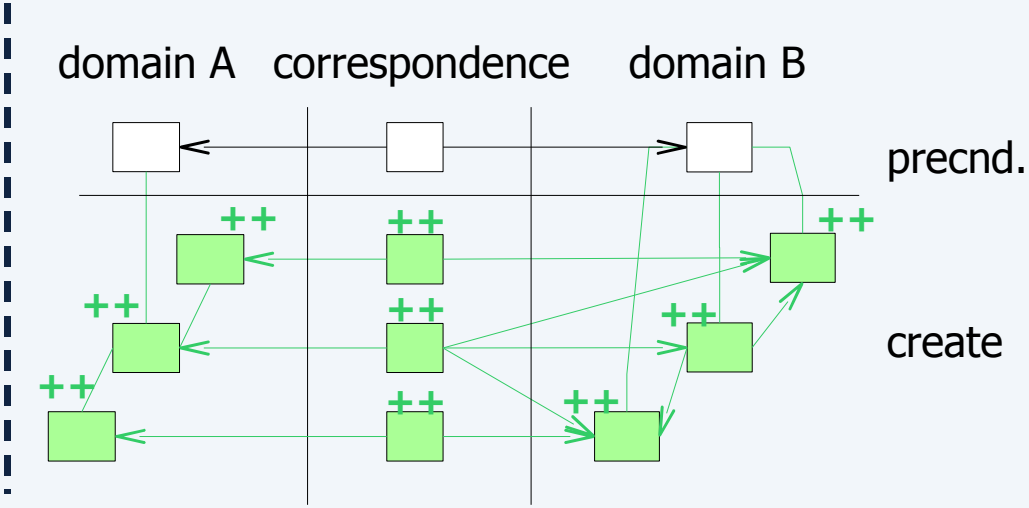
QVT-Core

```
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

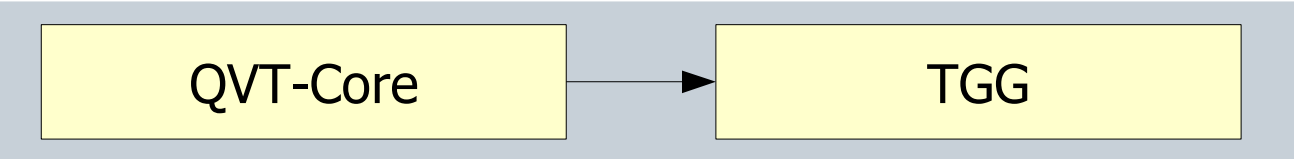


- Graph structure: Variables and OCL-Predicates
- One *Trace Class*

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

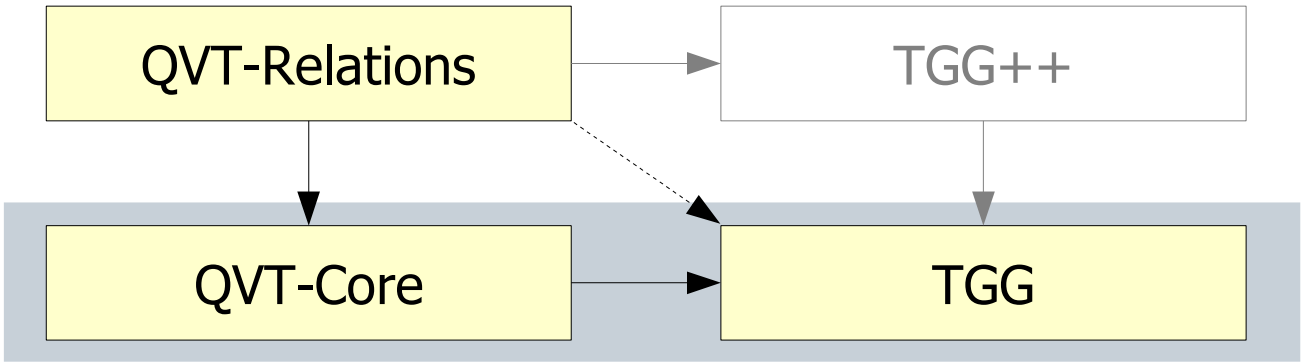
Mapping QVT to TGGs

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



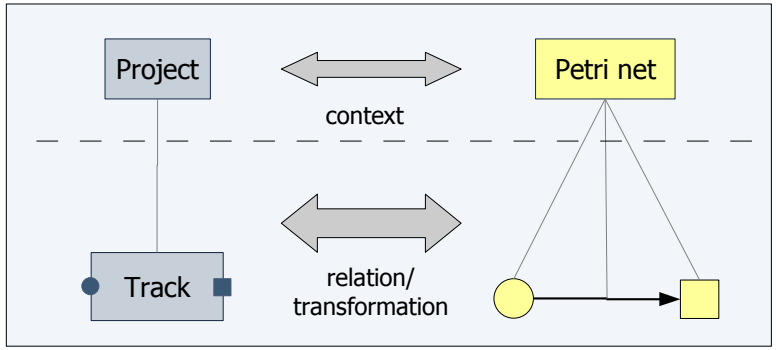
Mapping QVT to TGGs

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

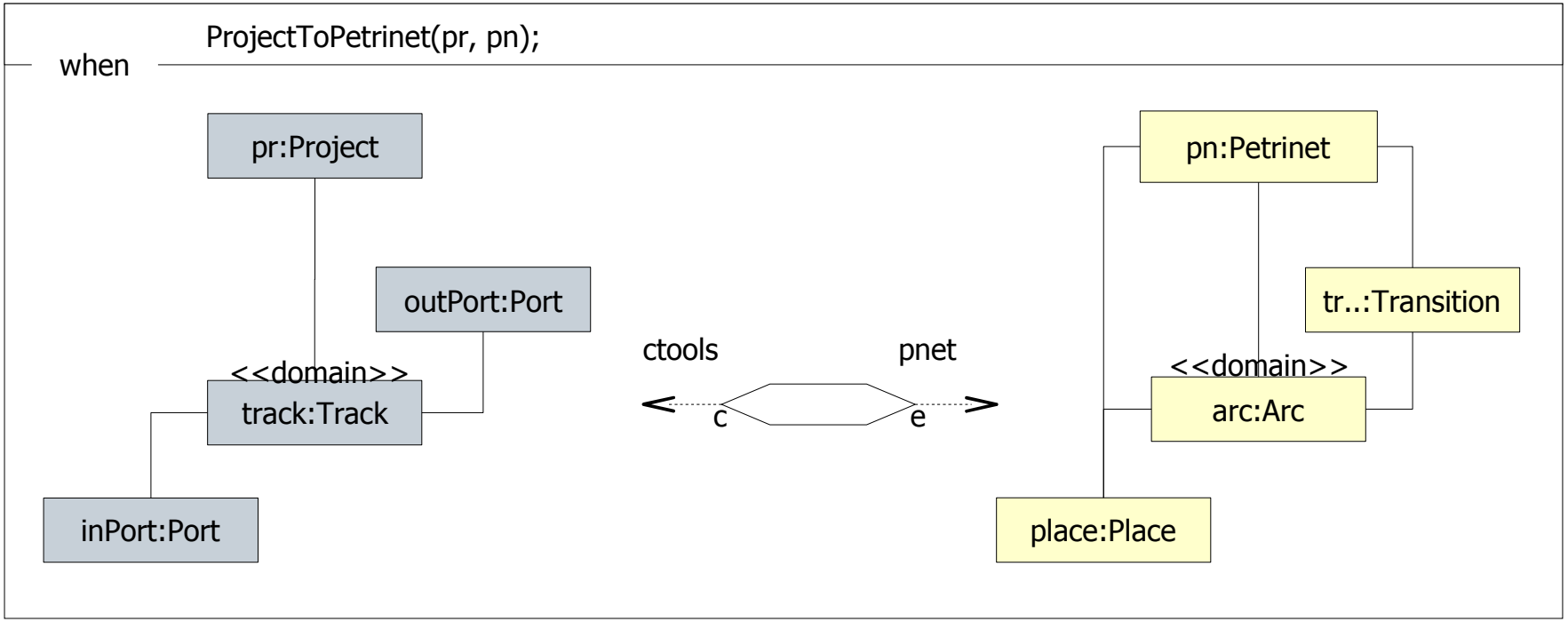


- Approach presented: QVT-Relations to TGG
- Learn from QVT: More usable "TGG++"?

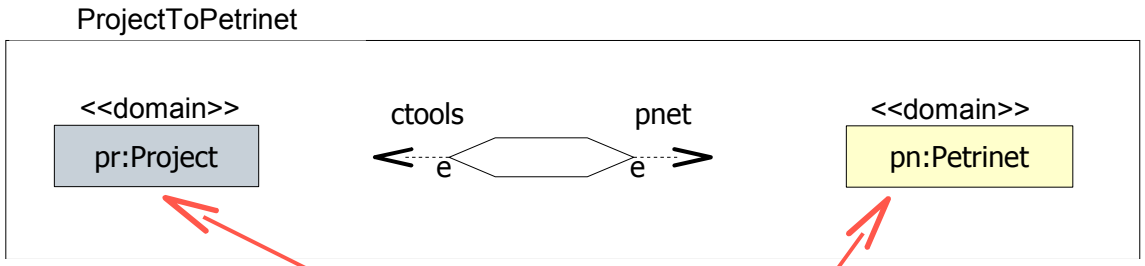
QVT Relation TrackToPlaceArcTransition

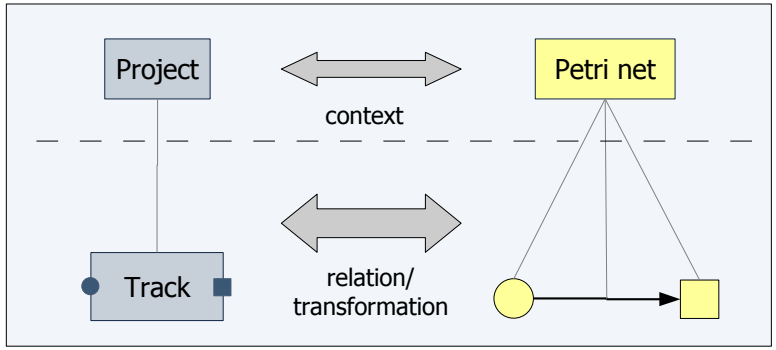


Relation TrackToPlaceArcTransition

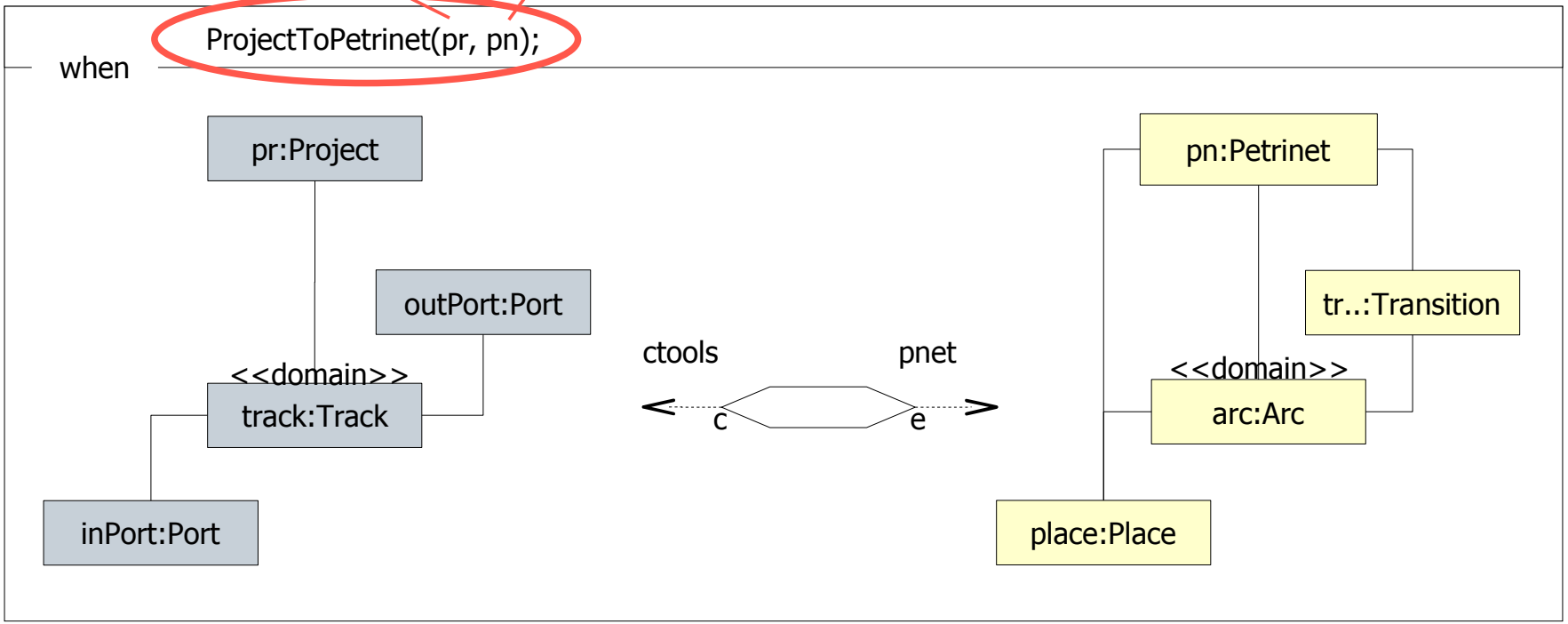


QVT Relation TrackToPlaceArcTransition

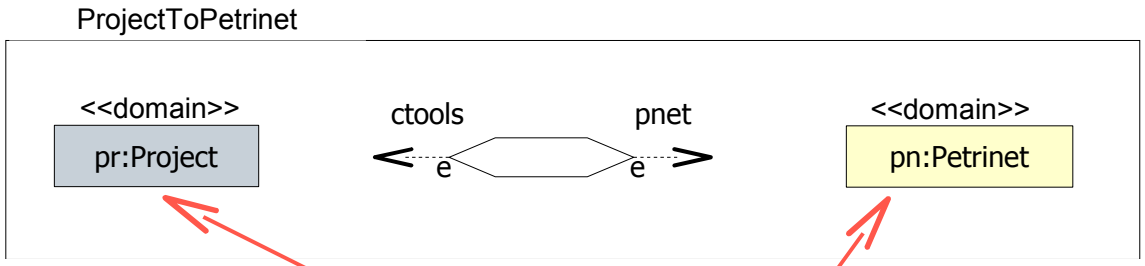


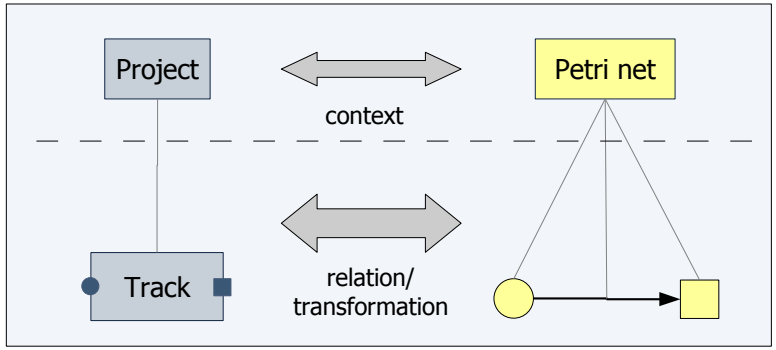


Relation TrackToPlaceArcTransition



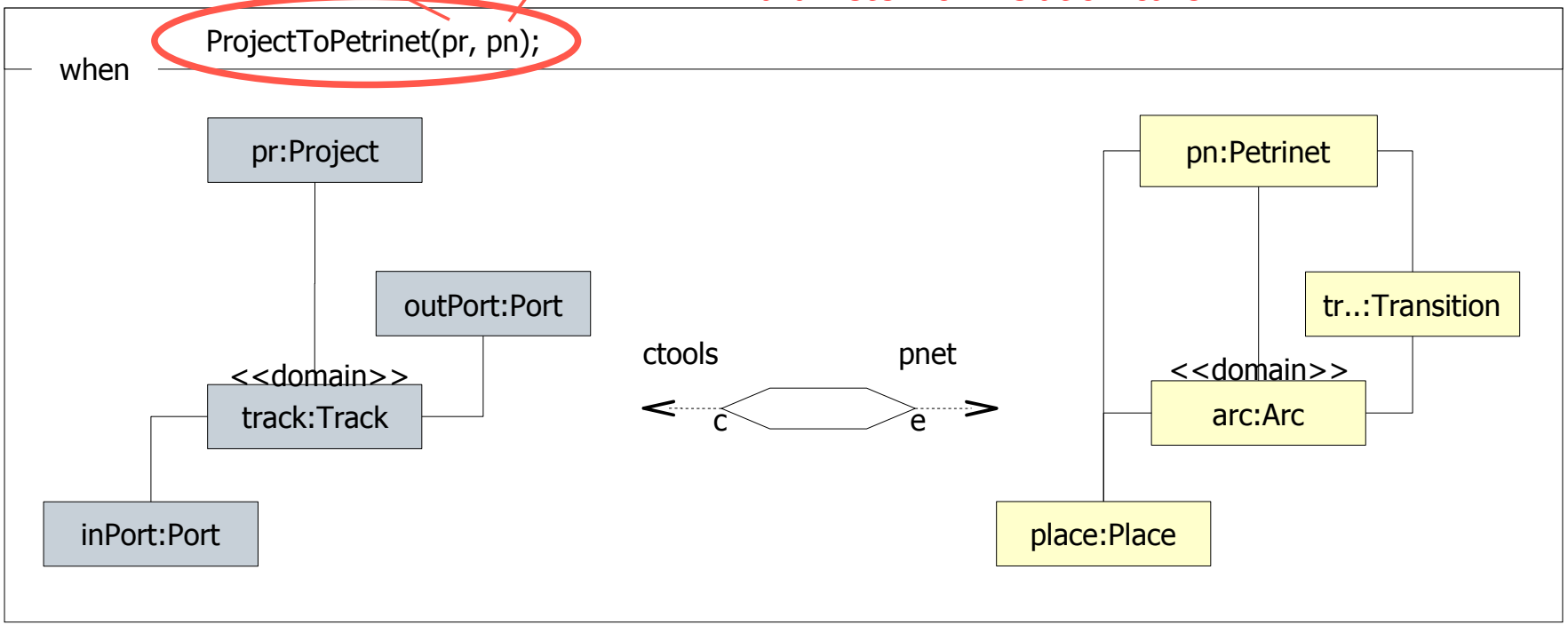
QVT Relation TrackToPlaceArcTransition





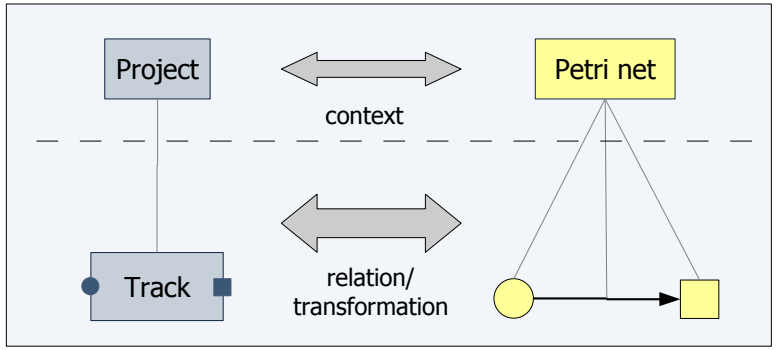
Domain-Element:
- "Root" of pattern
- Parameter for "relation calls"

Relation TrackToPlaceArcTransition

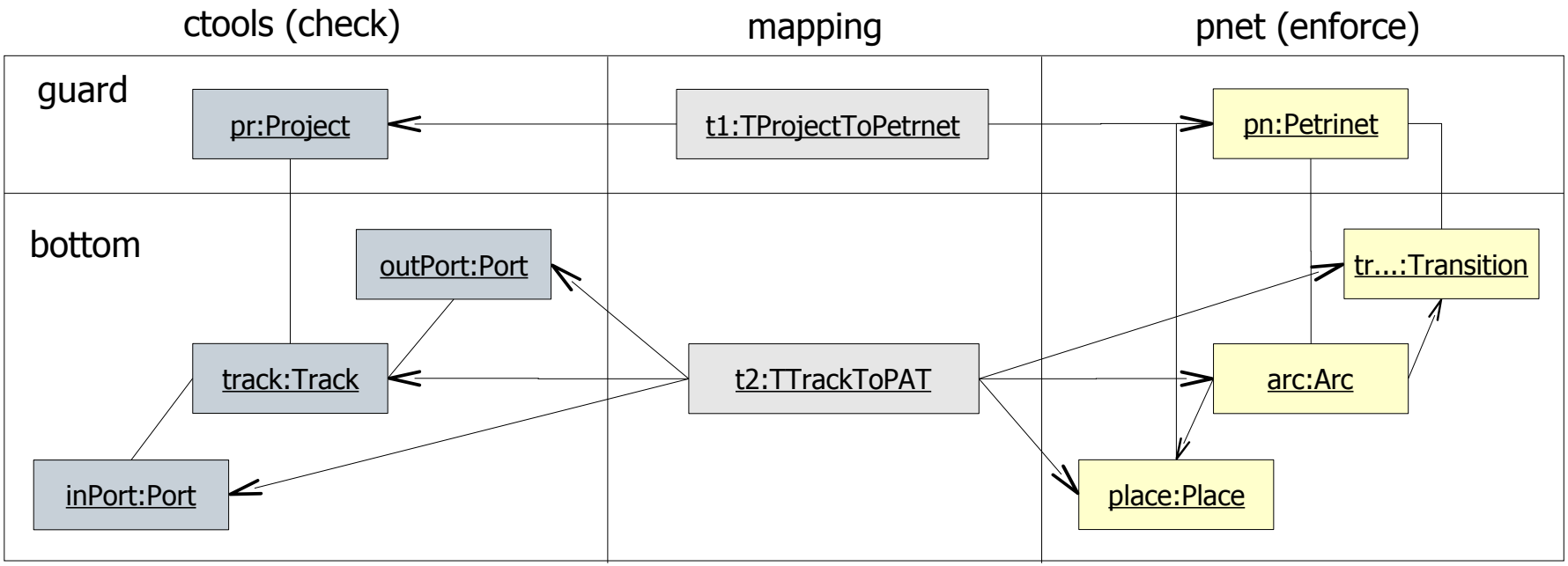


QVT-Core Mapping TrackToPlaceArcTransition

- Is transformed into a QVT-Core Mapping:

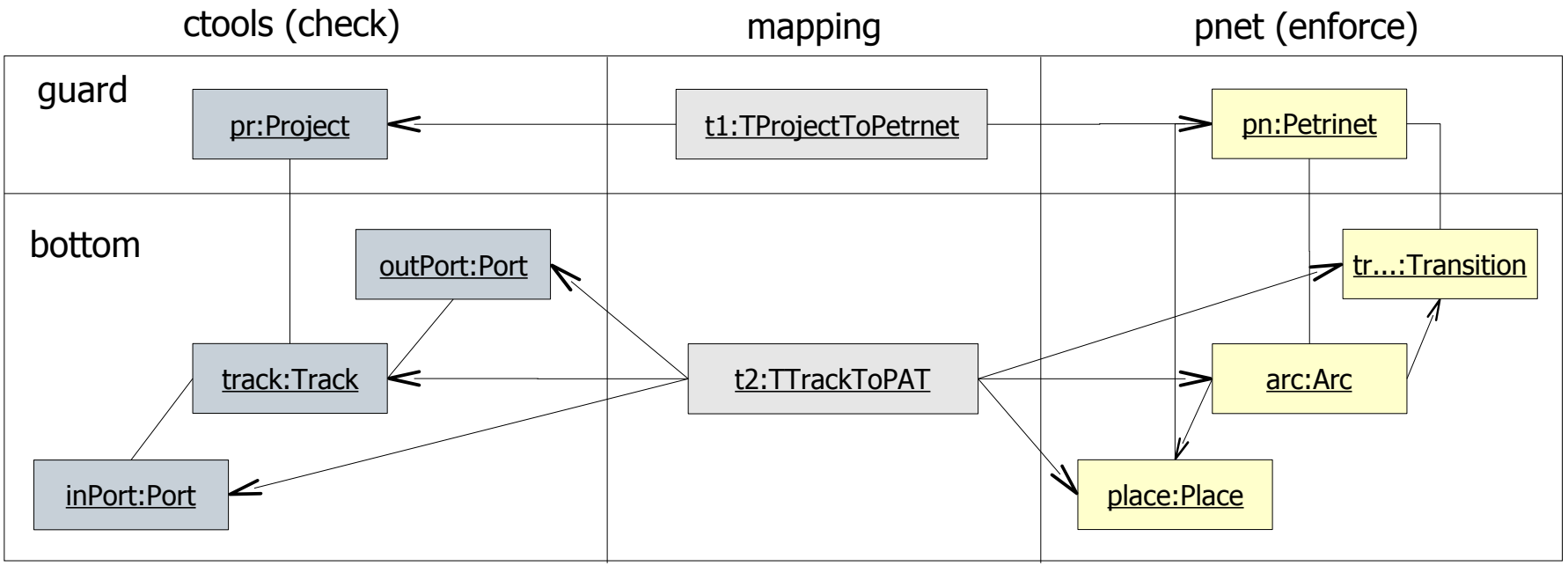


Mapping TrackToPlaceArcTransition



Semantics of a QVT-Core Mapping (fwd)

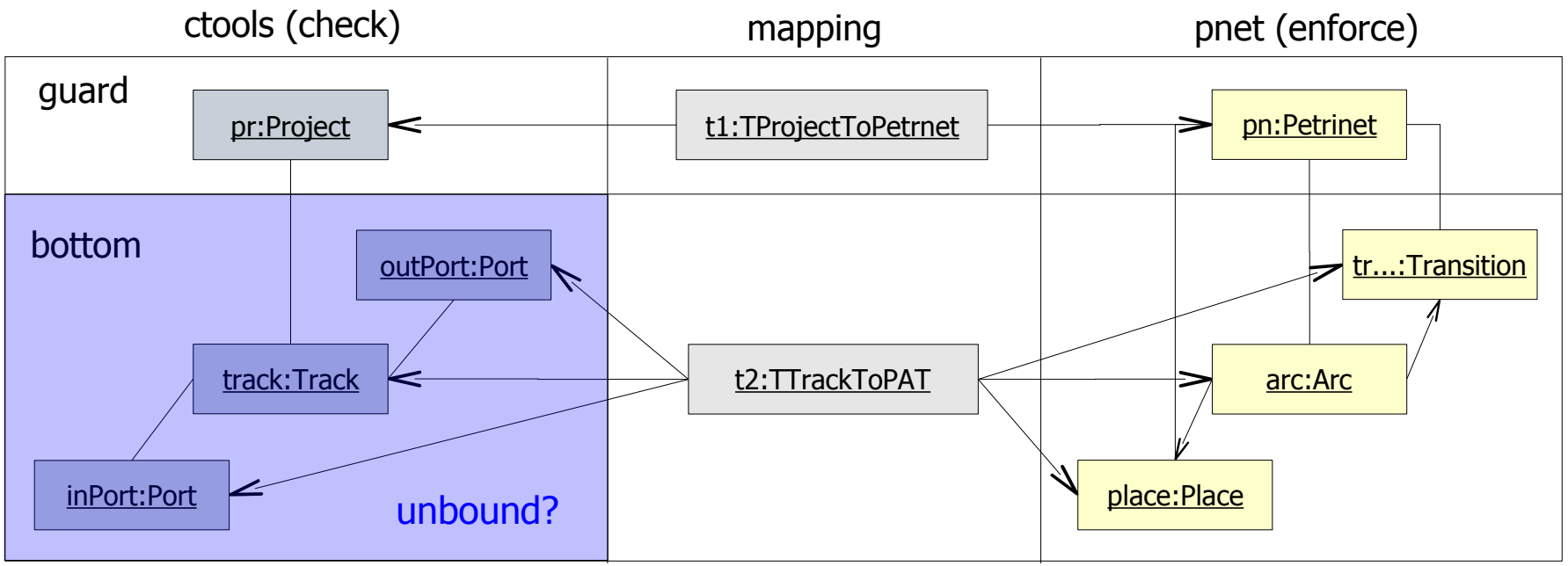
Mapping TrackToPlaceArcTransition



Semantics of a QVT-Core Mapping (fwd)

1) If Bottom Pattern found in unbound source model

Mapping TrackToPlaceArcTransition

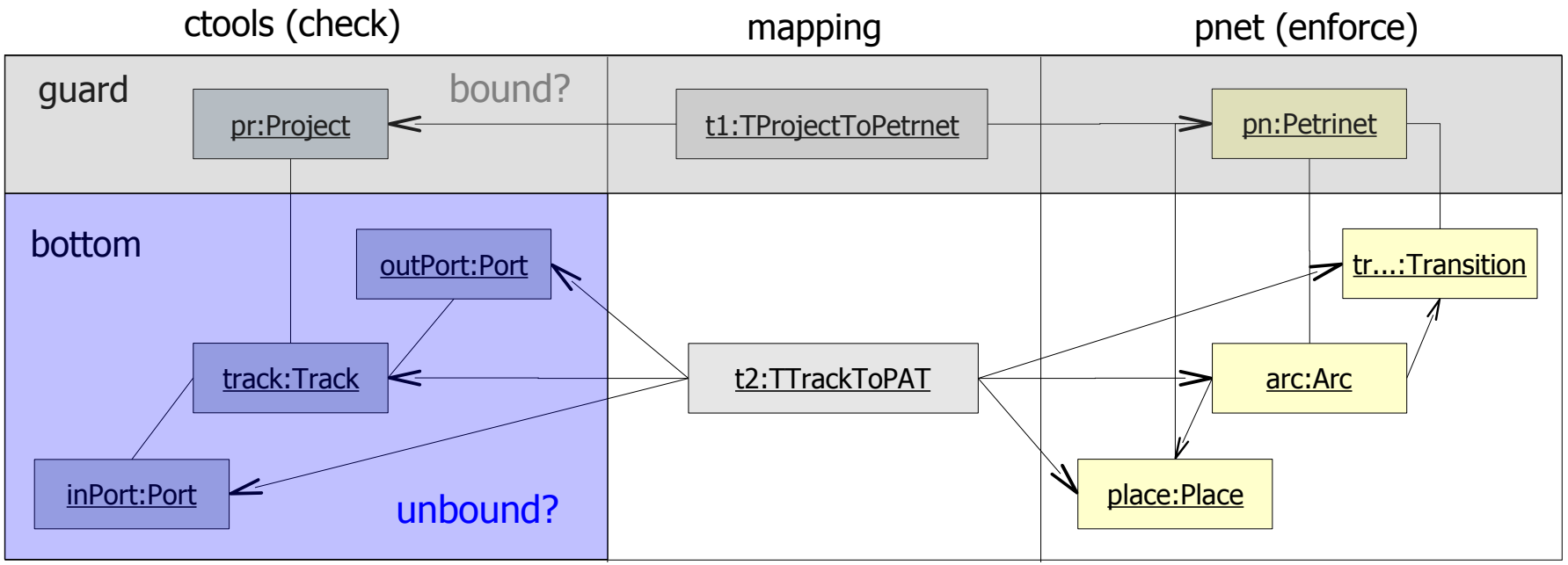


Semantics of a QVT-Core Mapping (fwd)

1) If Bottom Pattern found in unbound source model

2) & If Guard Pattern found in prev. bound model

Mapping TrackToPlaceArcTransition



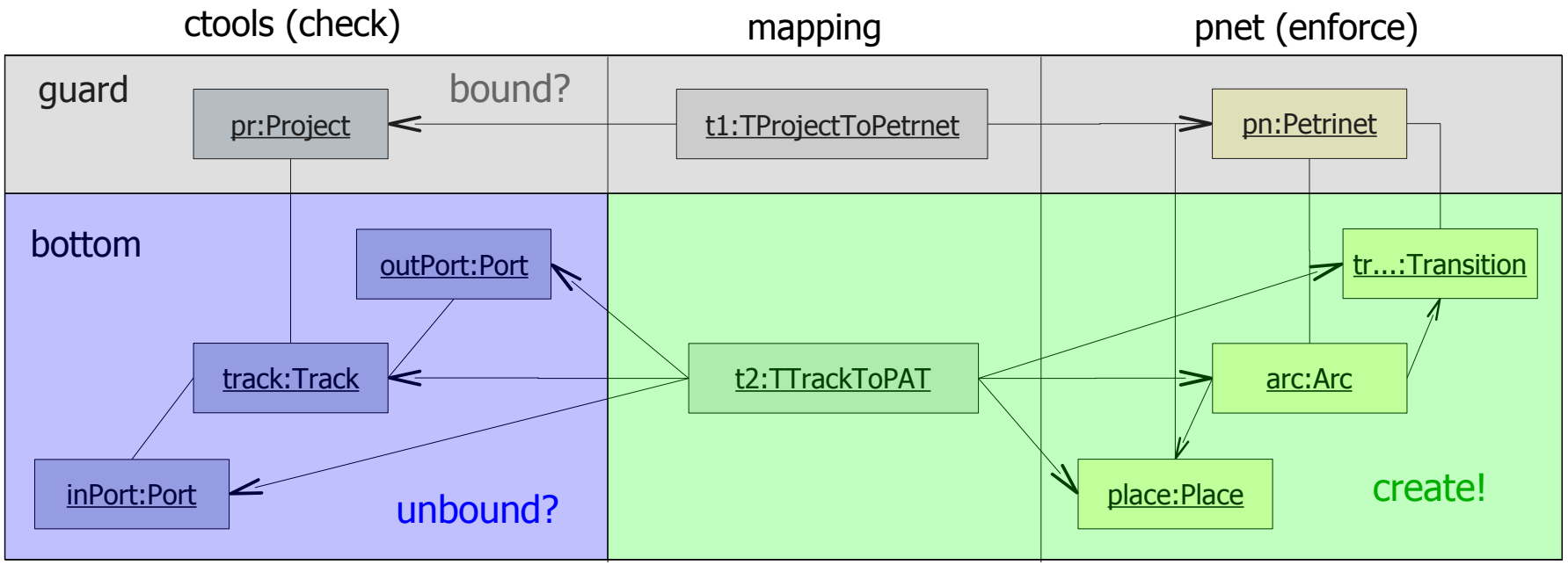
Semantics of a QVT-Core Mapping (fwd)

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)

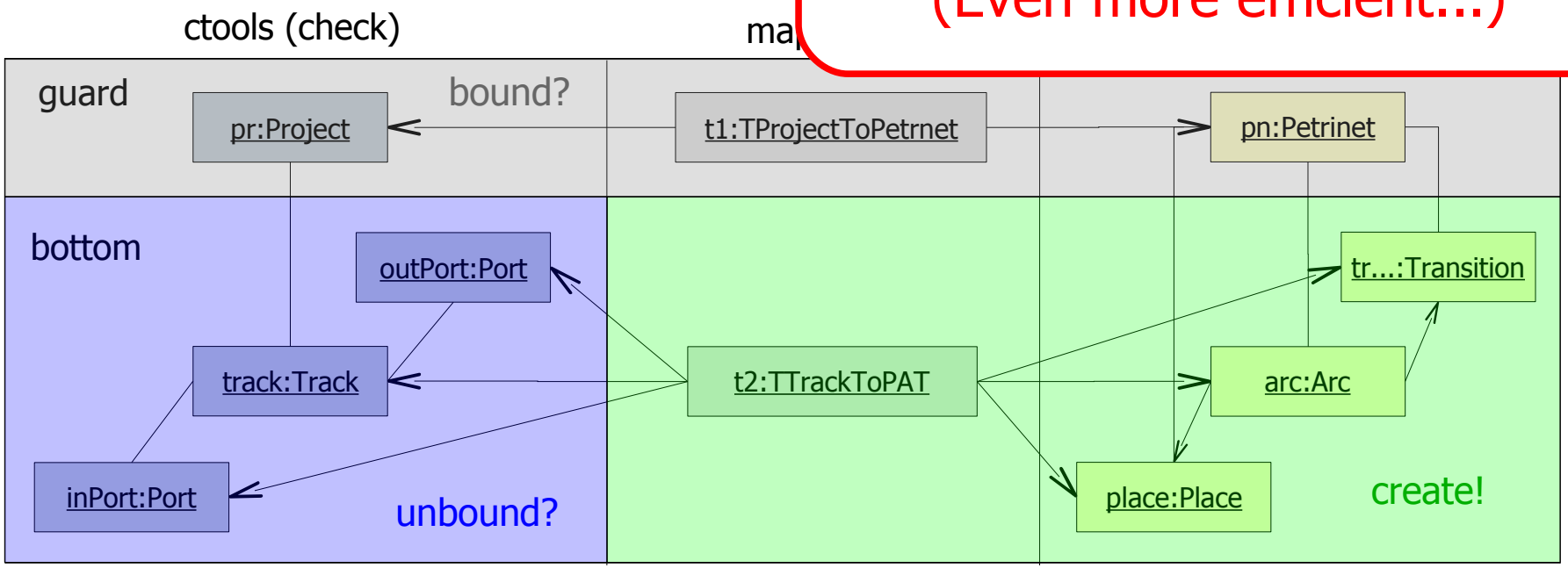
1) If Bottom Pattern found in unbound source model

2) & If Guard Pattern found in prev. bound model

3) Then create target mode

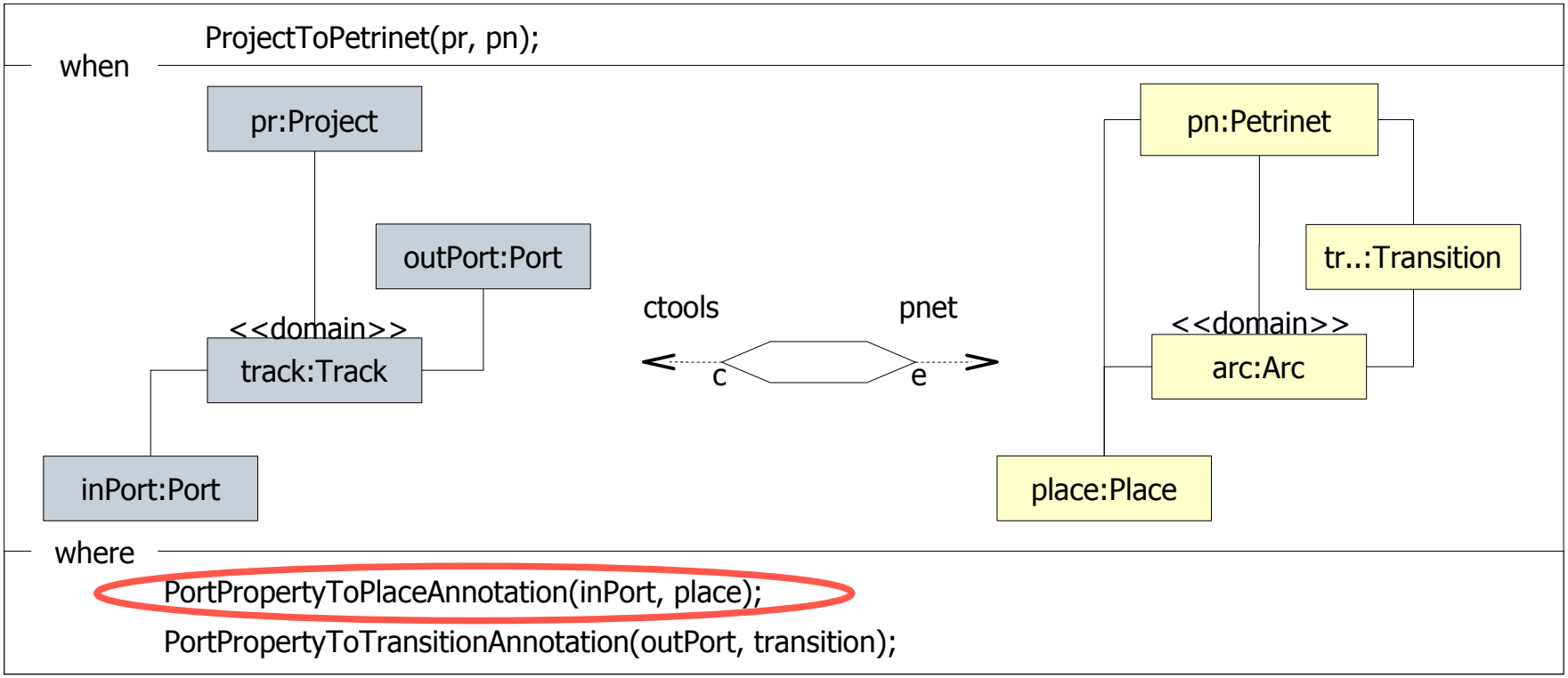
Different processing of TGGs,
but same result!
(Even more efficient...)

Mapping TrackToPlaceArcTransition



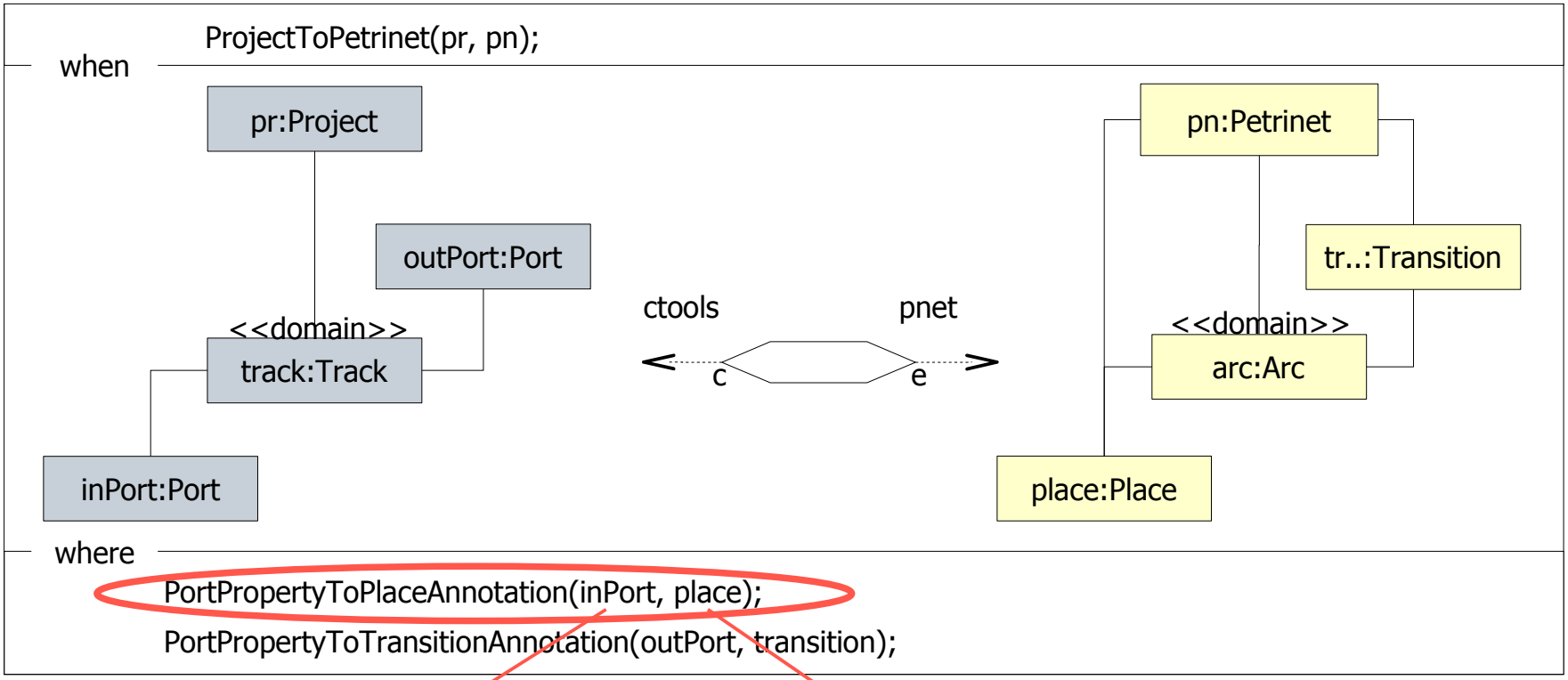
“where”-Relationships in QVT-Relations

Relation TrackToPlaceArcTransition

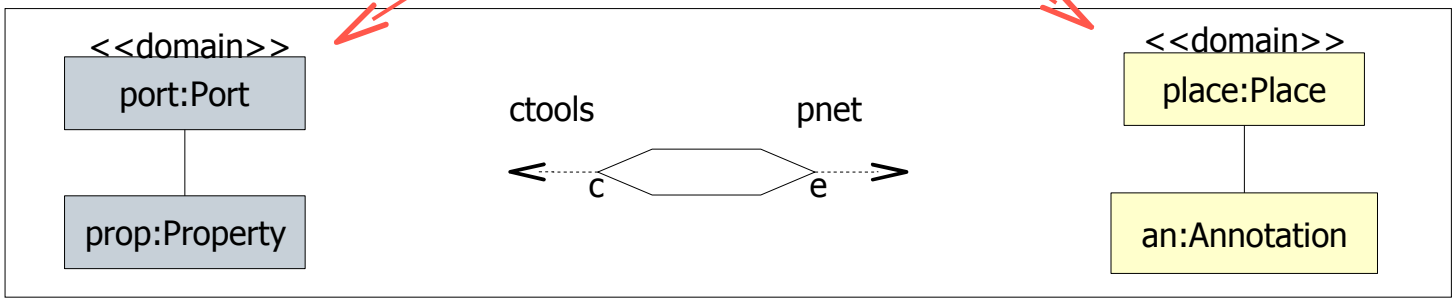


“where”-Relationships in QVT-Relations

Relation TrackToPlaceArcTransition

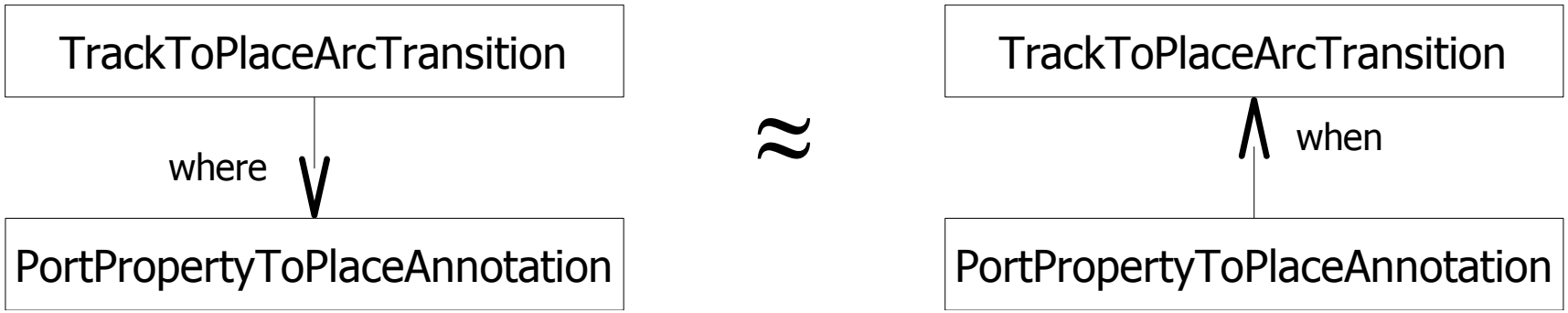


Relation TrackToPlaceArcTransition



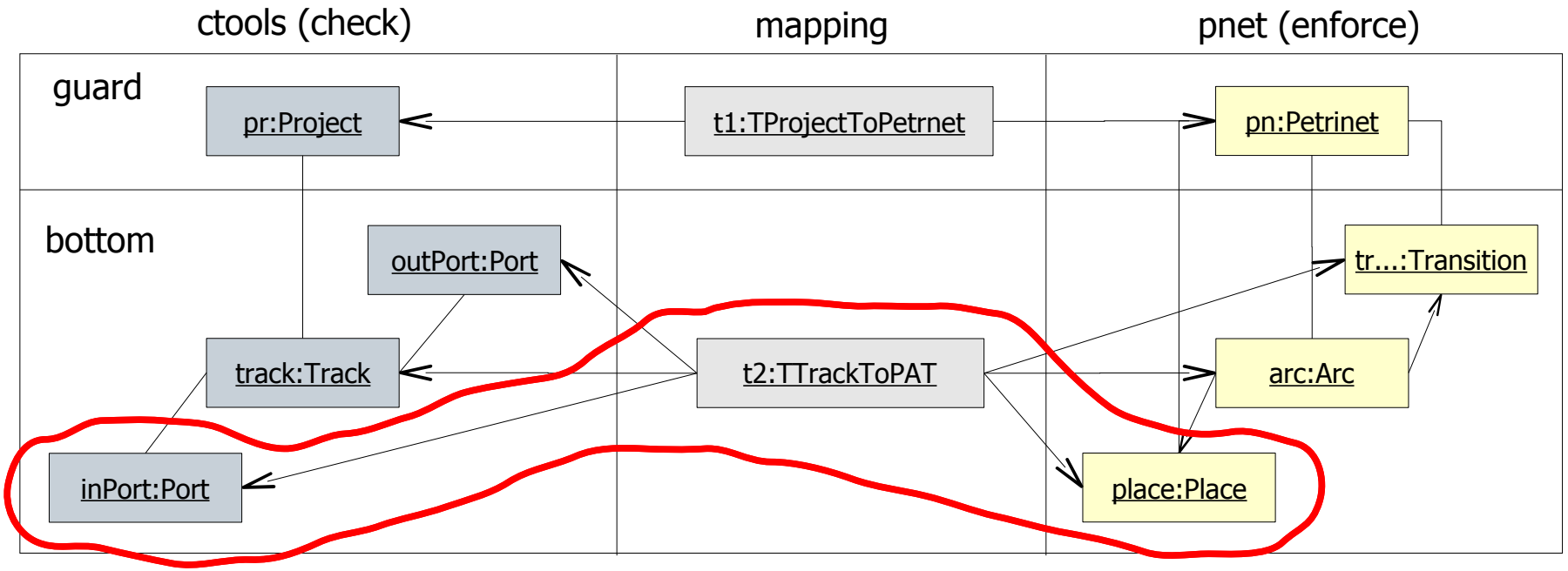
When vs. Where

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

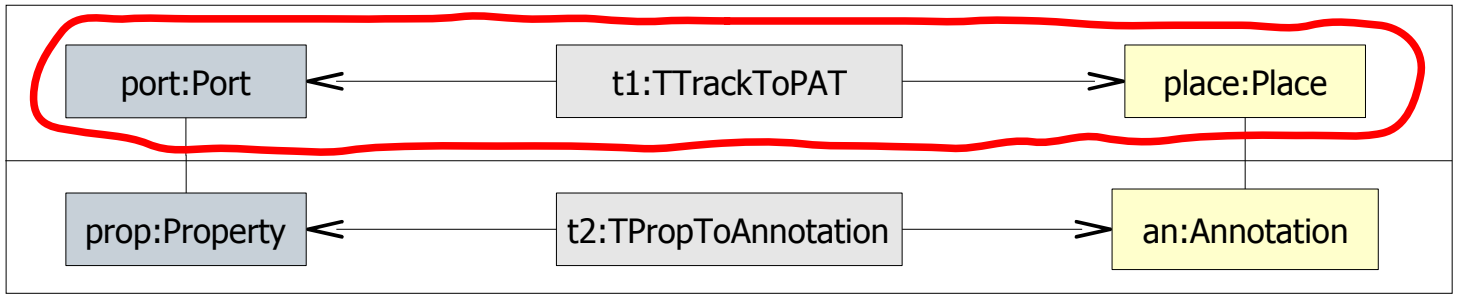


A "When"-Relationship in QVT-Core

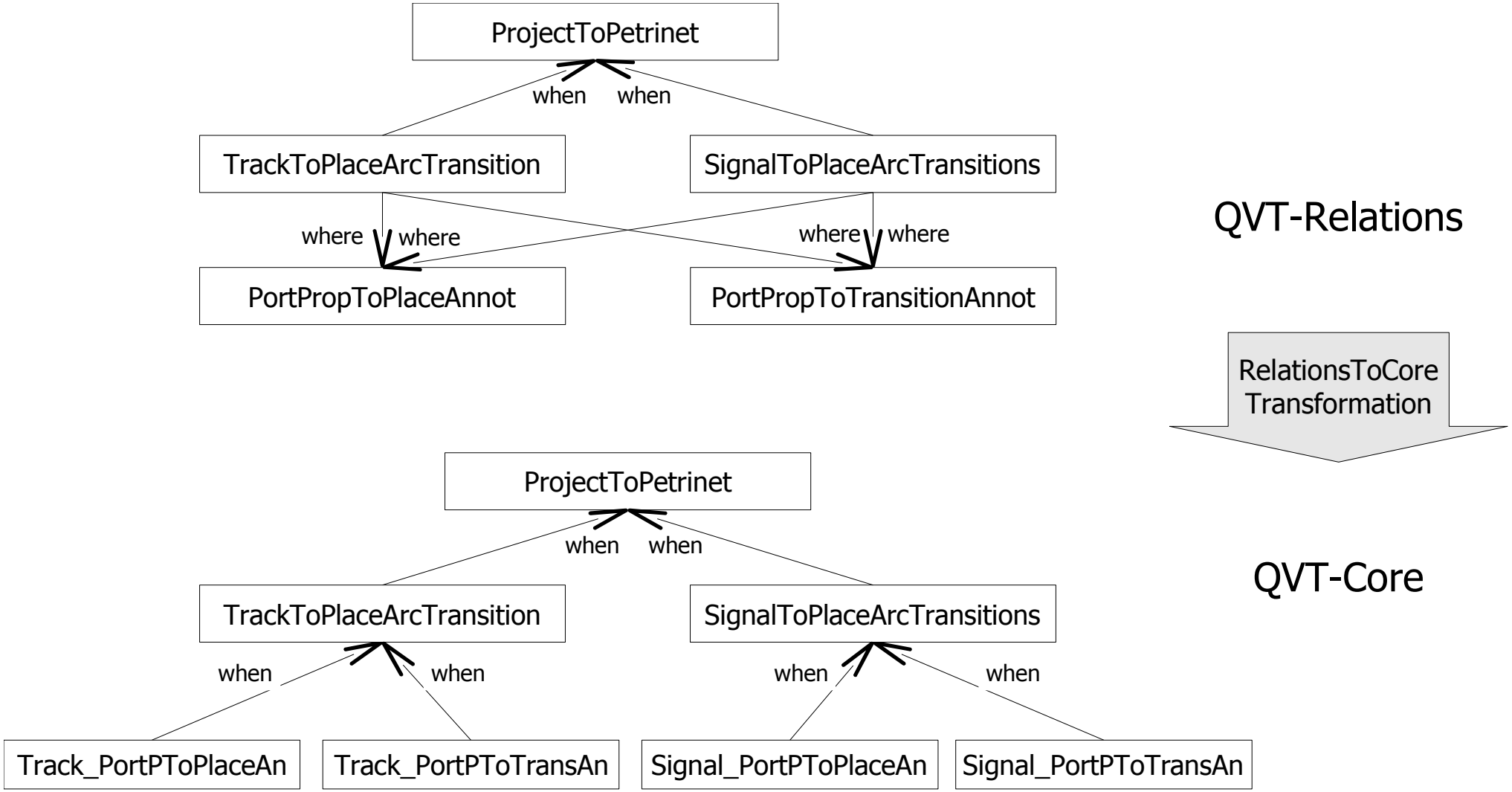
Mapping TrackToPlaceArcTransition



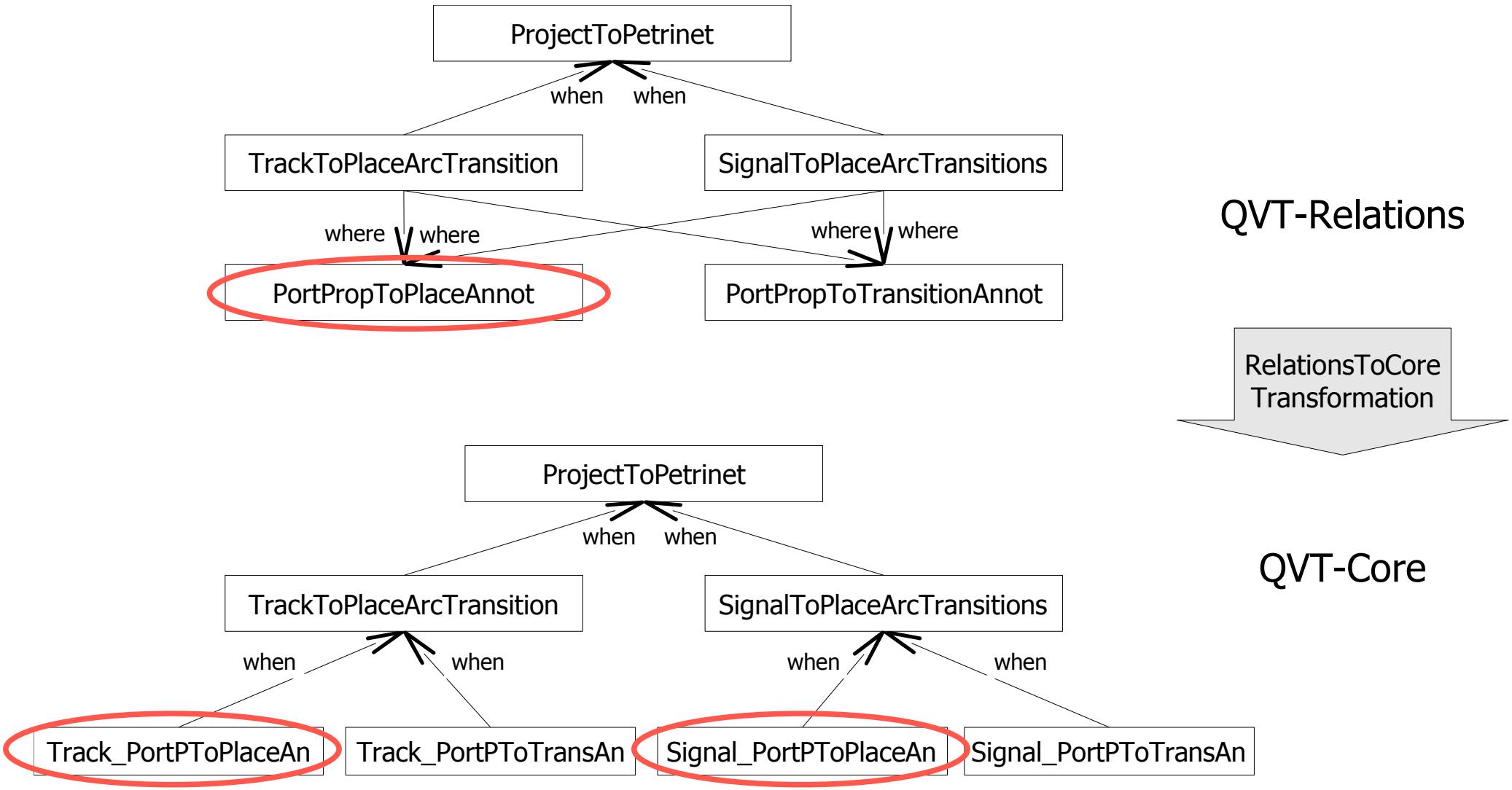
Mapping PortPropertyToPlaceAnnotation



when&where Relationships in QVT-Core

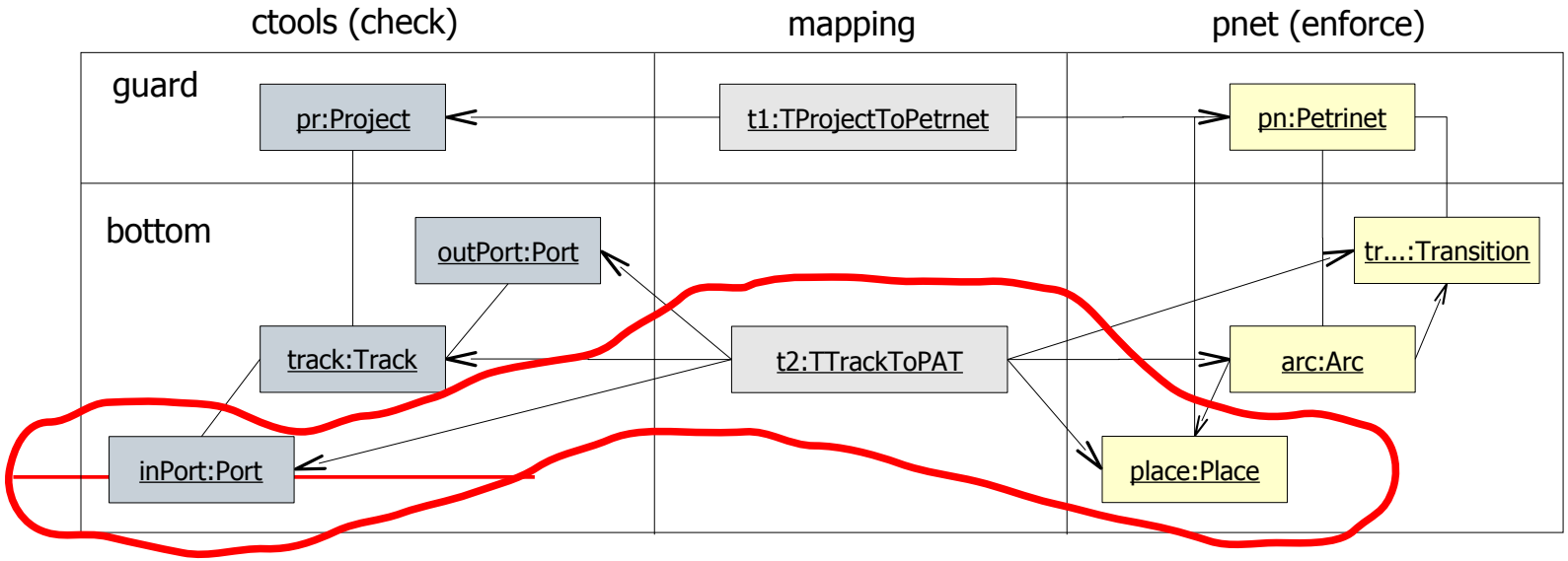


when&where Relationships in QVT-Core

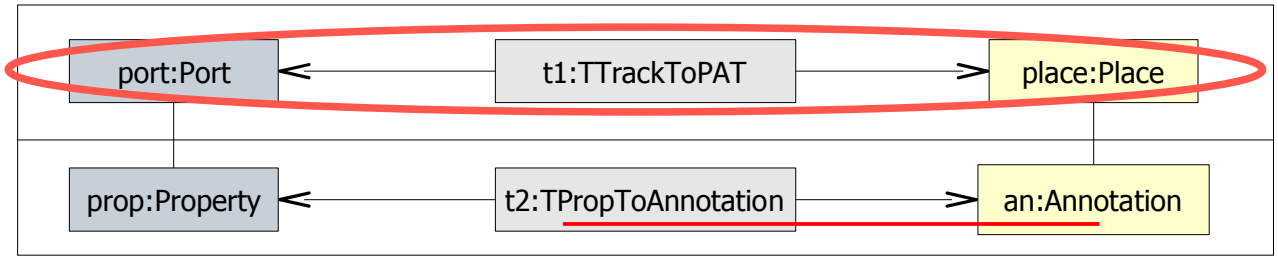


“Unfolded” Mappings

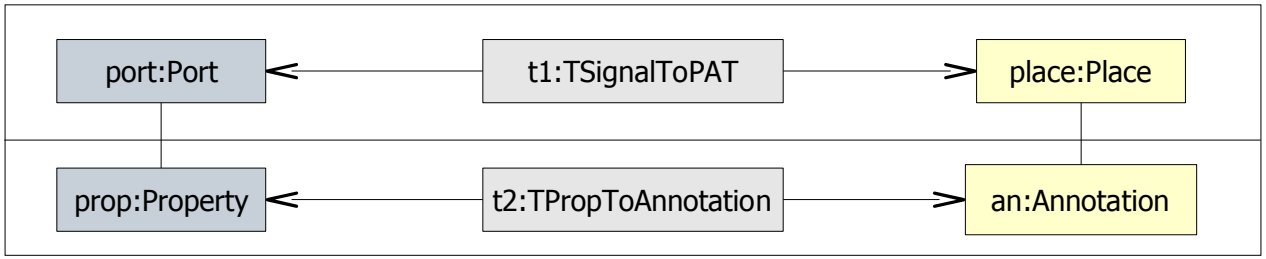
Mapping TrackToPlaceArcTransition



Mapping TrackToPlaceArcTransition_PortPropertyToPlaceAnnotation

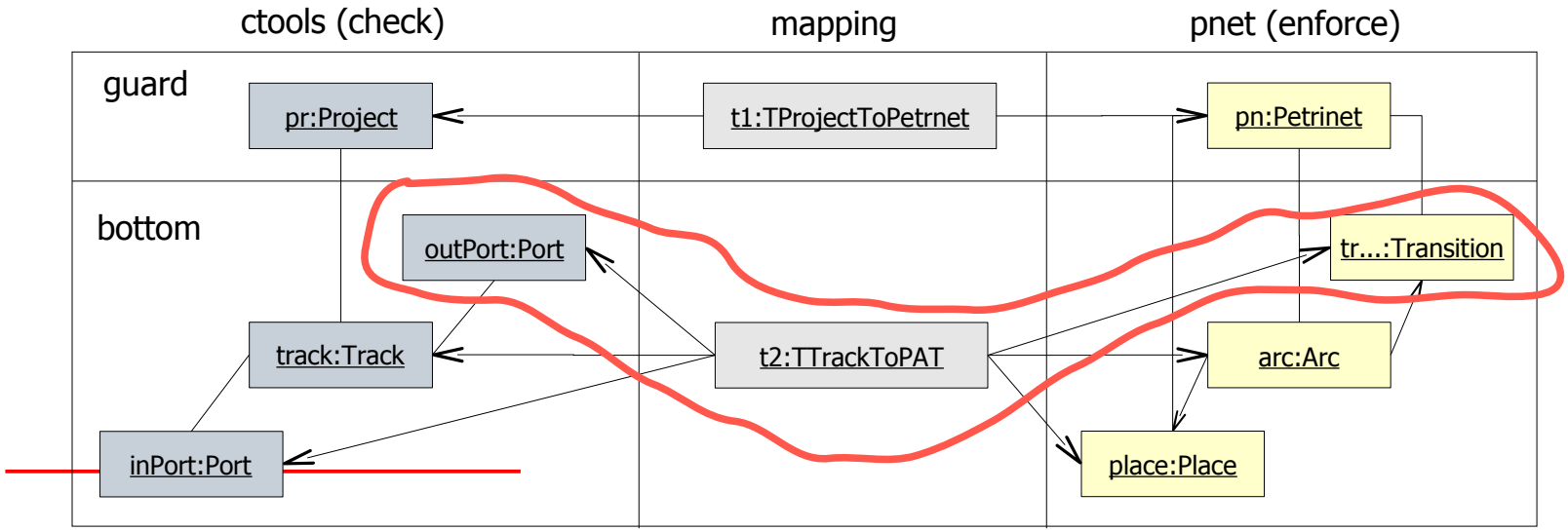


Mapping SignalToPlaceArcTransition_PortPropertyToPlaceAnnotation

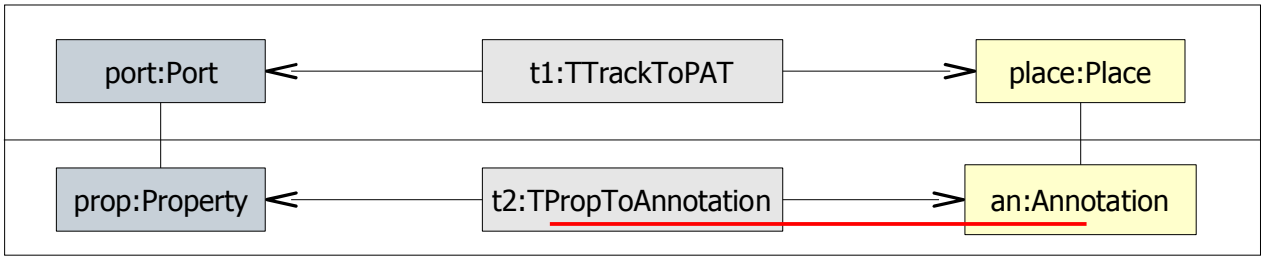


"Unfolded" Mappings

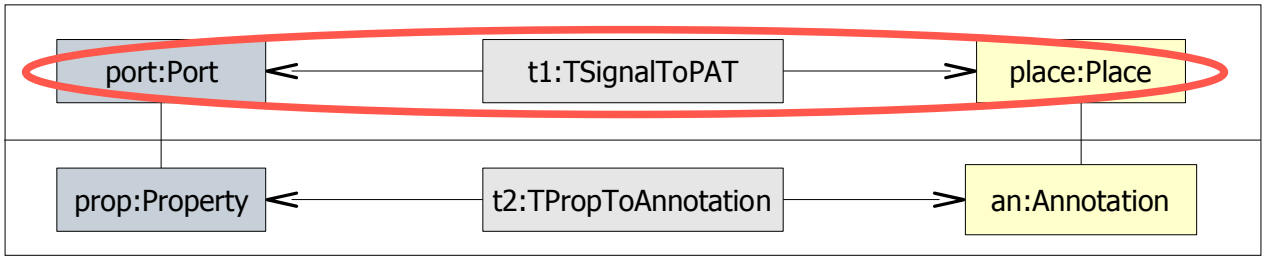
Mapping TrackToPlaceArcTransition



Mapping TrackToPlaceArcTransition_PortPropertyToPlaceAnnotation

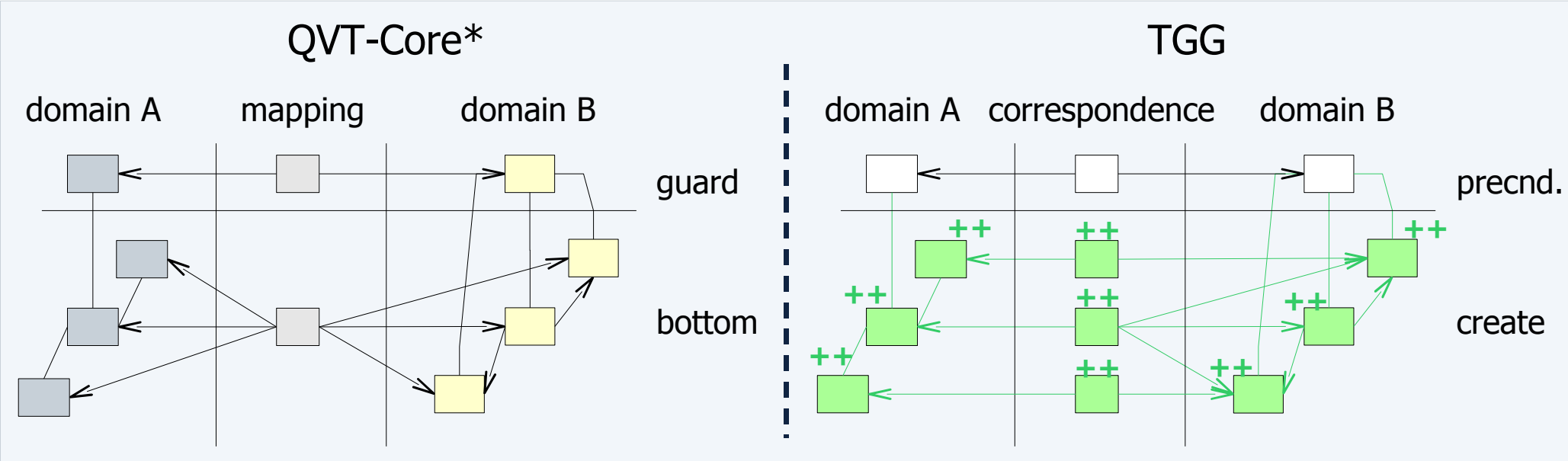


Mapping SignalToPlaceArcTransition_PortPropertyToPlaceAnnotation



QVT-Core vs. TGGs – Short Comparison

- Structurally similar rules



- 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

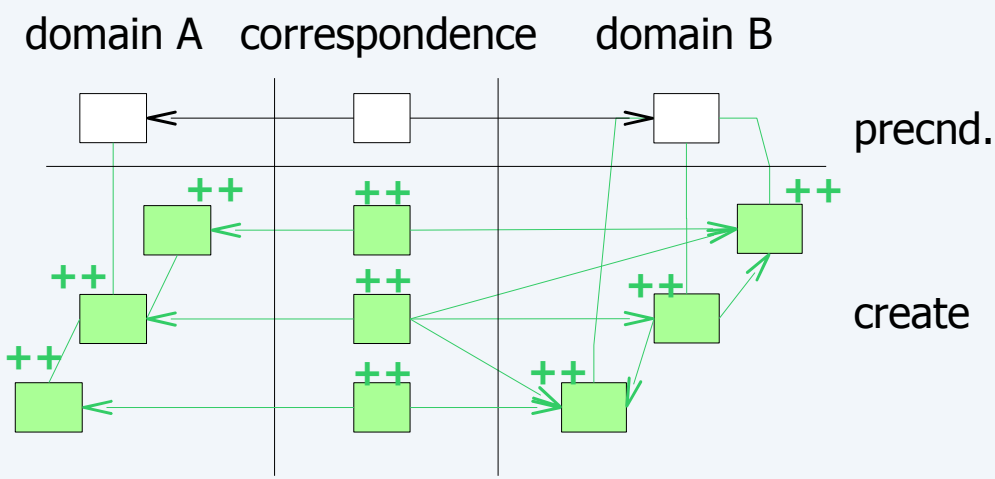
QVT-Core

```
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



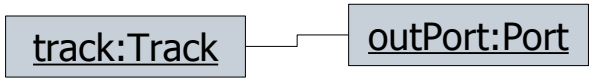
- Graph structure: Variables and OCL-Predicates
- One *Trace Class*

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

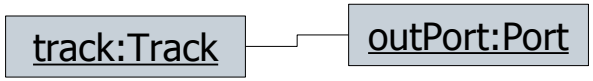
QVT-Core vs. TGGs – Short Comparison

QVT-Core	TGG
Variable	Node

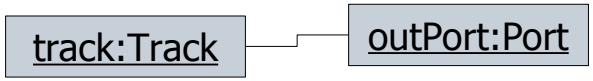
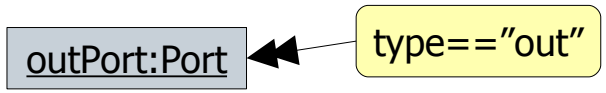
QVT-Core vs. TGGs – Short Comparison

QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 

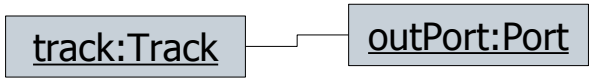
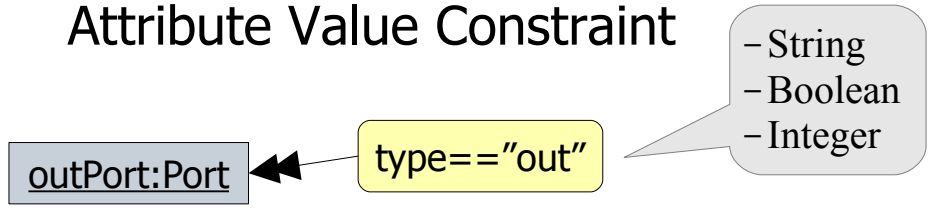
QVT-Core vs. TGGs – Short Comparison

QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 
(Simple) Attribute Value Predicate or Attribute Value Assignment	Attribute Value Constraint

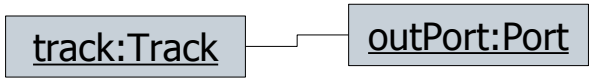
QVT-Core vs. TGGs – Short Comparison

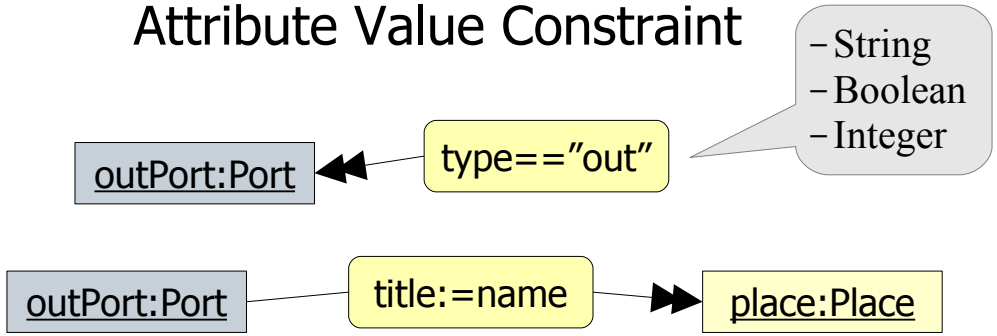
QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 
(Simple) Attribute Value Predicate or Attribute Value Assignment `outPort.type=="out";`	Attribute Value Constraint 

QVT-Core vs. TGGs – Short Comparison

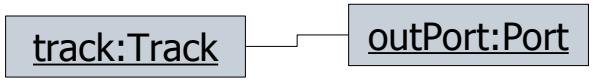
QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 
(Simple) Attribute Value Predicate or Attribute Value Assignment `outPort.type=="out";`	Attribute Value Constraint 

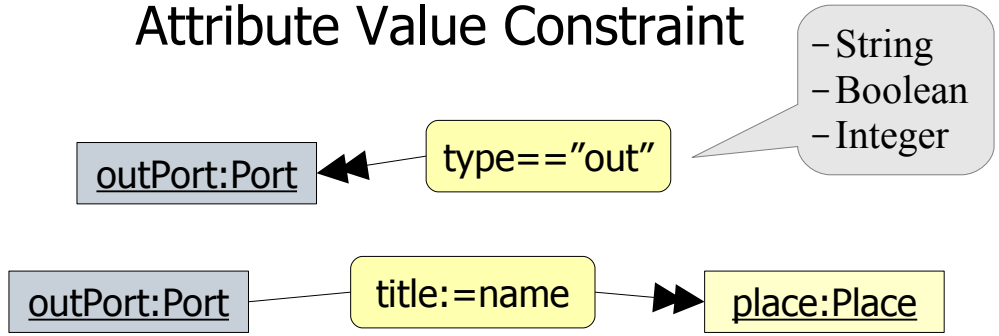
QVT-Core vs. TGGs – Short Comparison

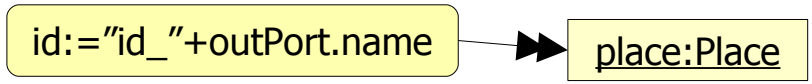
QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 

(Simple) Attribute Value Predicate or Attribute Value Assignment
`outPort.type=="out";`
`place.title:=outPort.name;`
	Attribute Value Constraint
	 

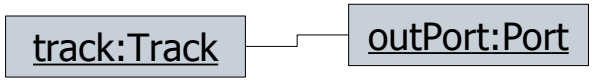
QVT-Core vs. TGGs – Short Comparison

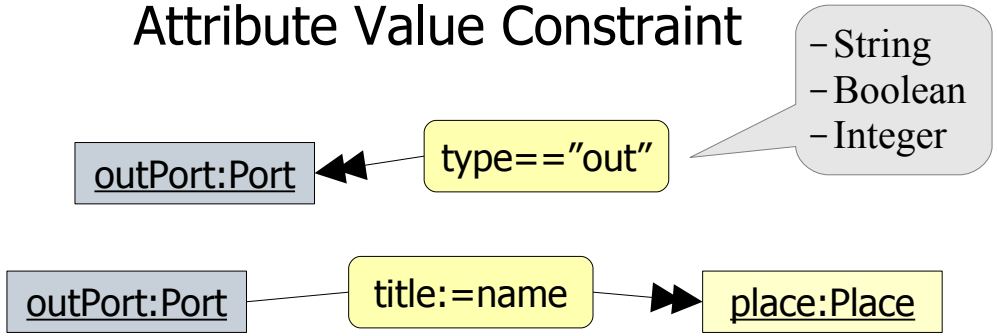
QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 

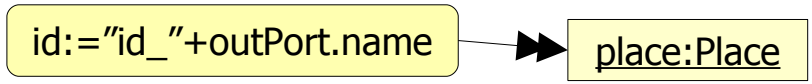
(Simple) Attribute Value Predicate or Attribute Value Assignment
`outPort.type=="out";`
`place.title:=outPort.name;`
	Attribute Value Constraint
	 

(Complex) Attribute Value Predicate/Assignment `place.id:="id_"+outPort.name;`	OCL Attribute Value Constraint 

QVT-Core vs. TGGs – Short Comparison

QVT-Core	TGG
Variable	Node
Reference Predicate (==) or Reference Assignment (:=) `track.port:=outPort;`	Edge 

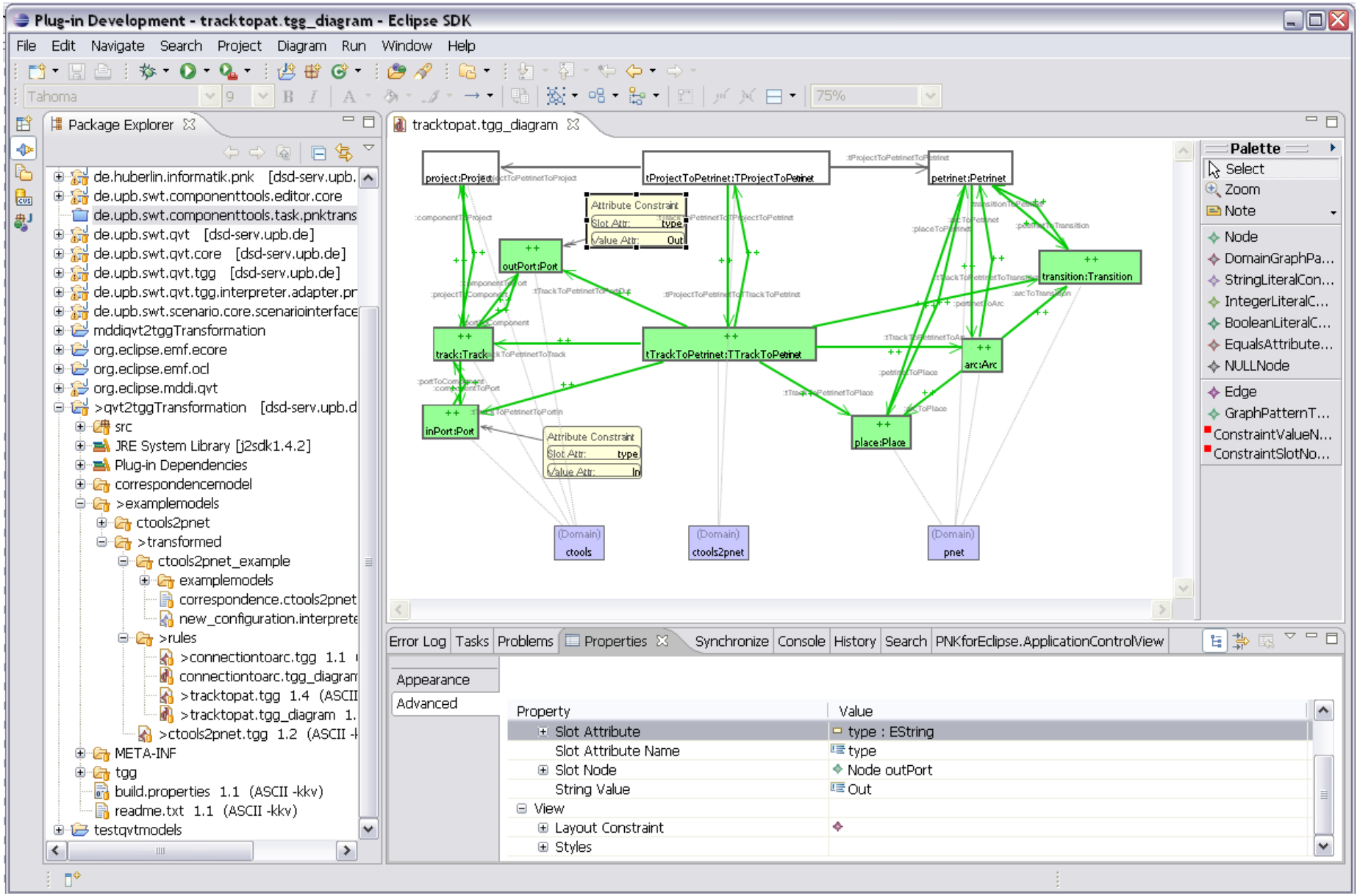
(Simple) Attribute Value Predicate or Attribute Value Assignment
`outPort.type=="out";`
`place.title:=outPort.name;`
	Attribute Value Constraint
	 

(Complex) Attribute Value Predicate/Assignment `place.id:="id_"+outPort.name;`	OCL Attribute Value Constraint Outlook 

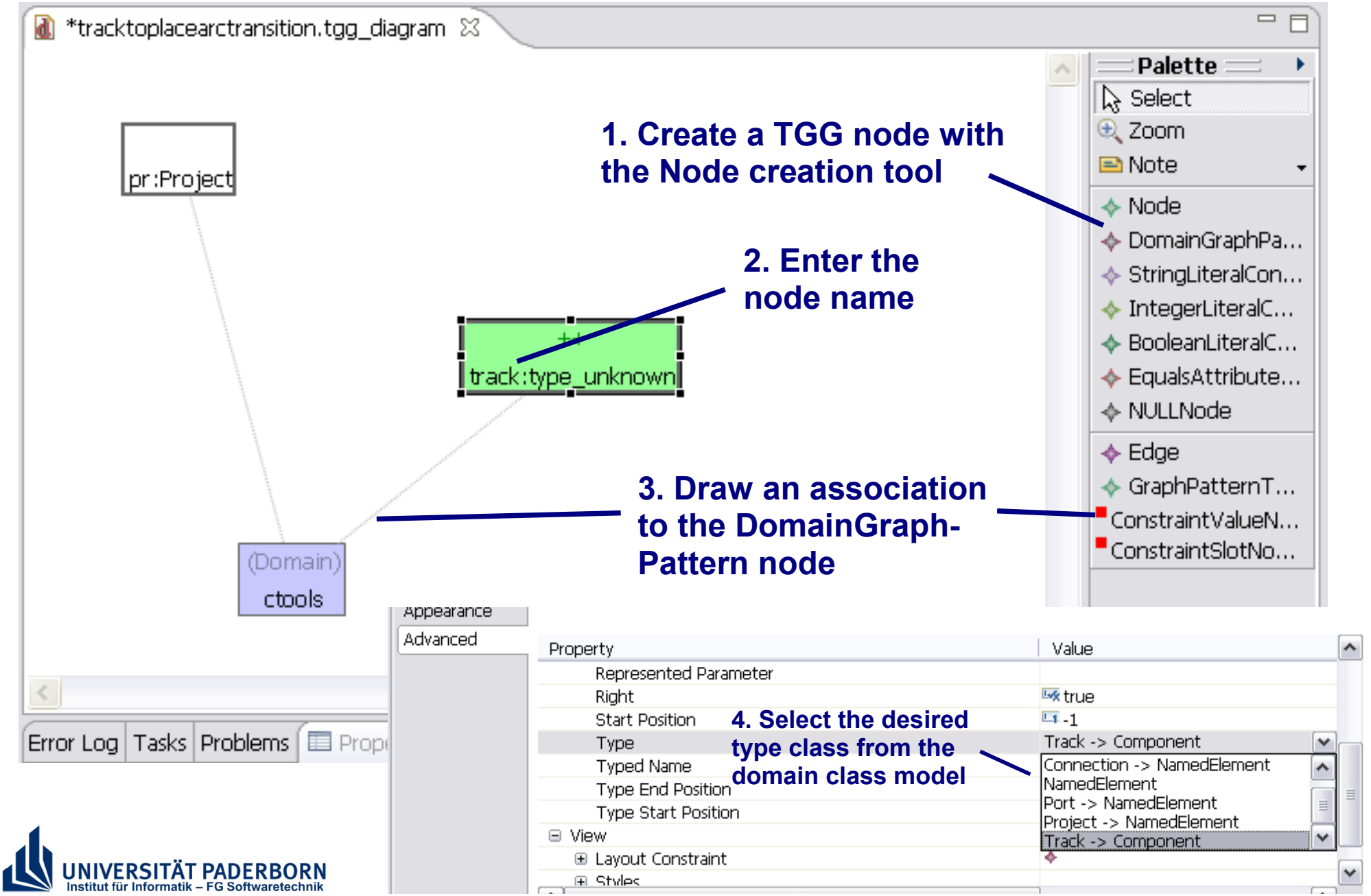
Summary

- 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 valueable input for QVT

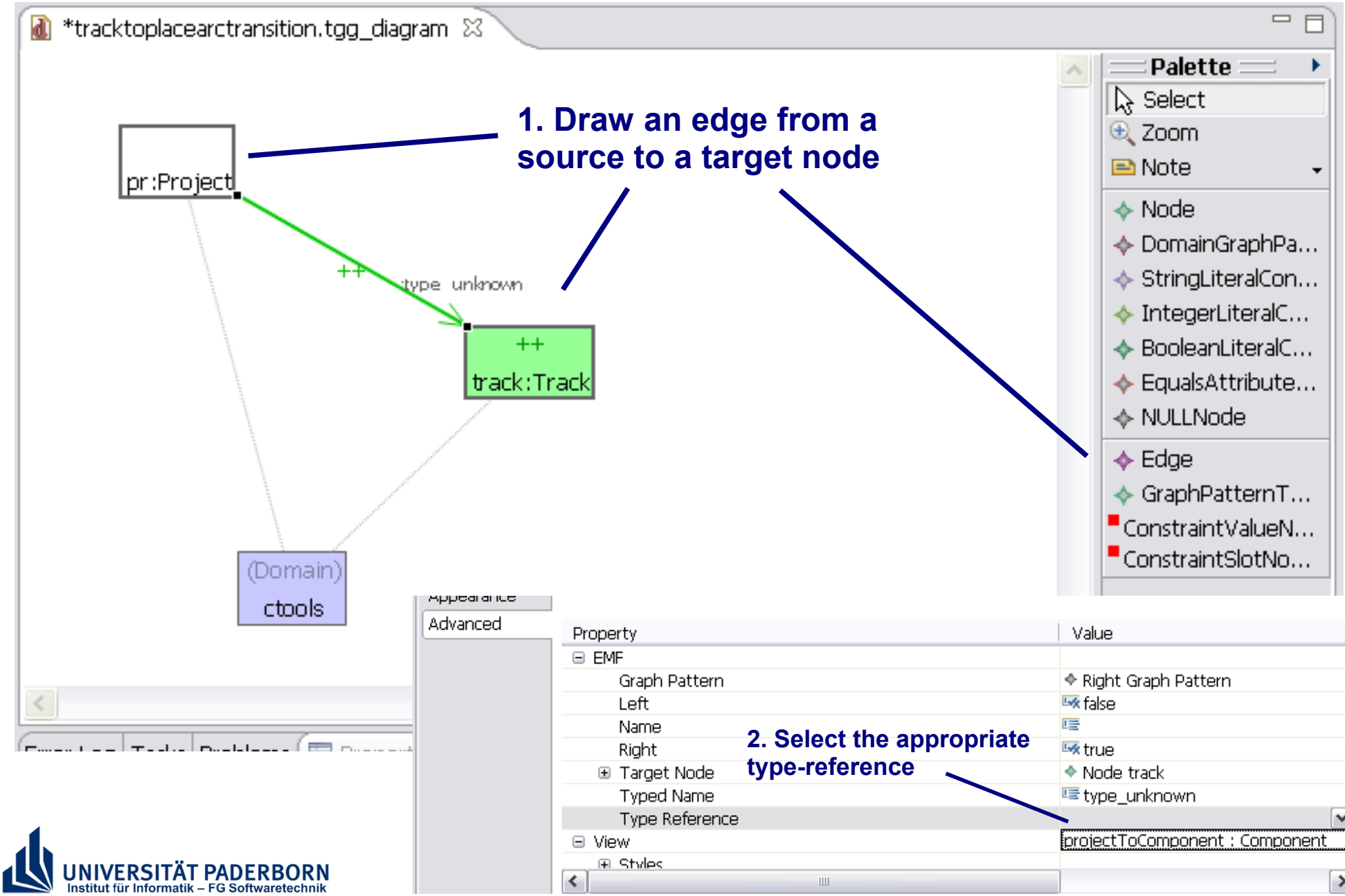
TGG Editor Screenshot



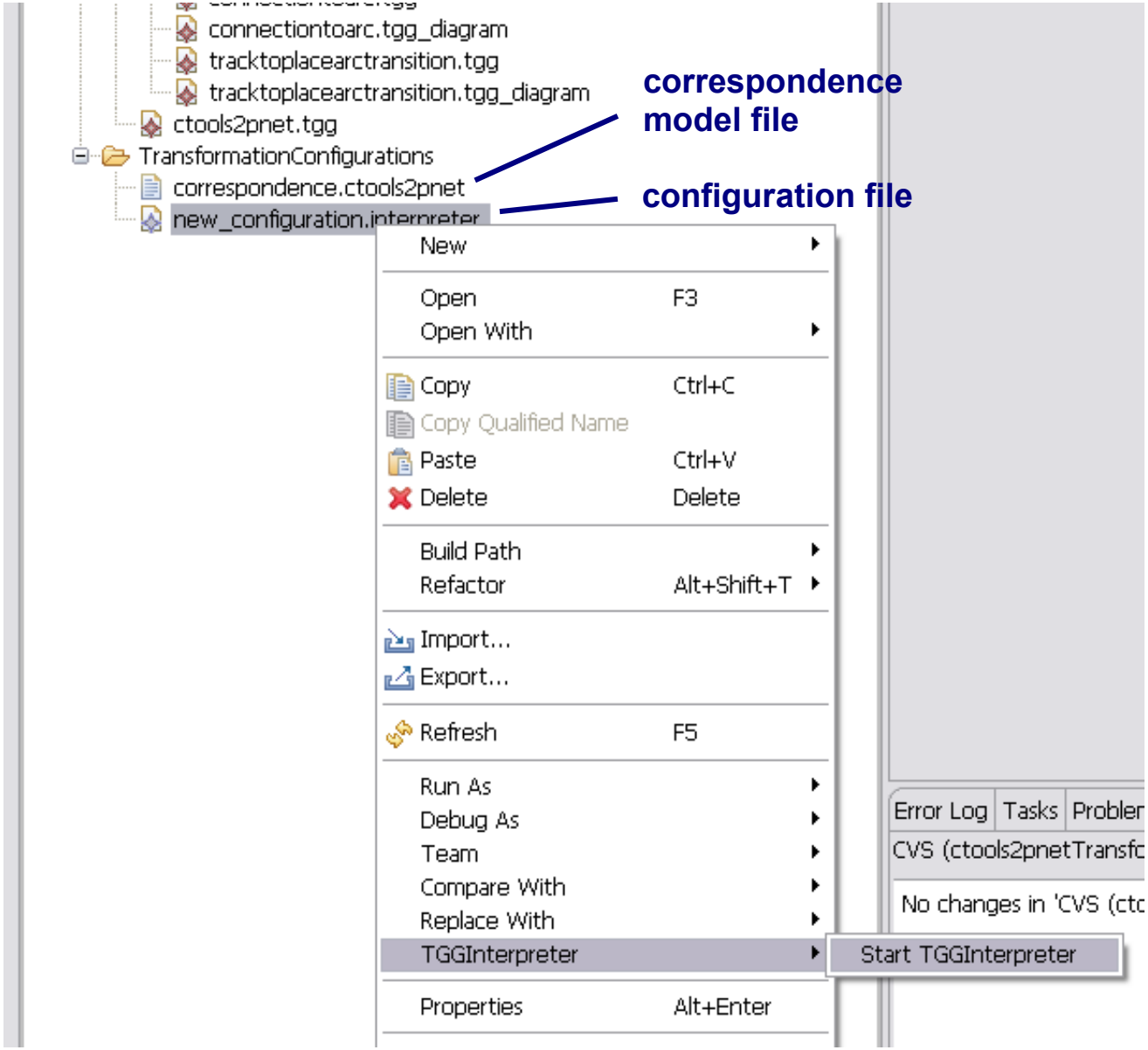
TGG Editor – Creating a Node



TGG Editor – Creating an Edge

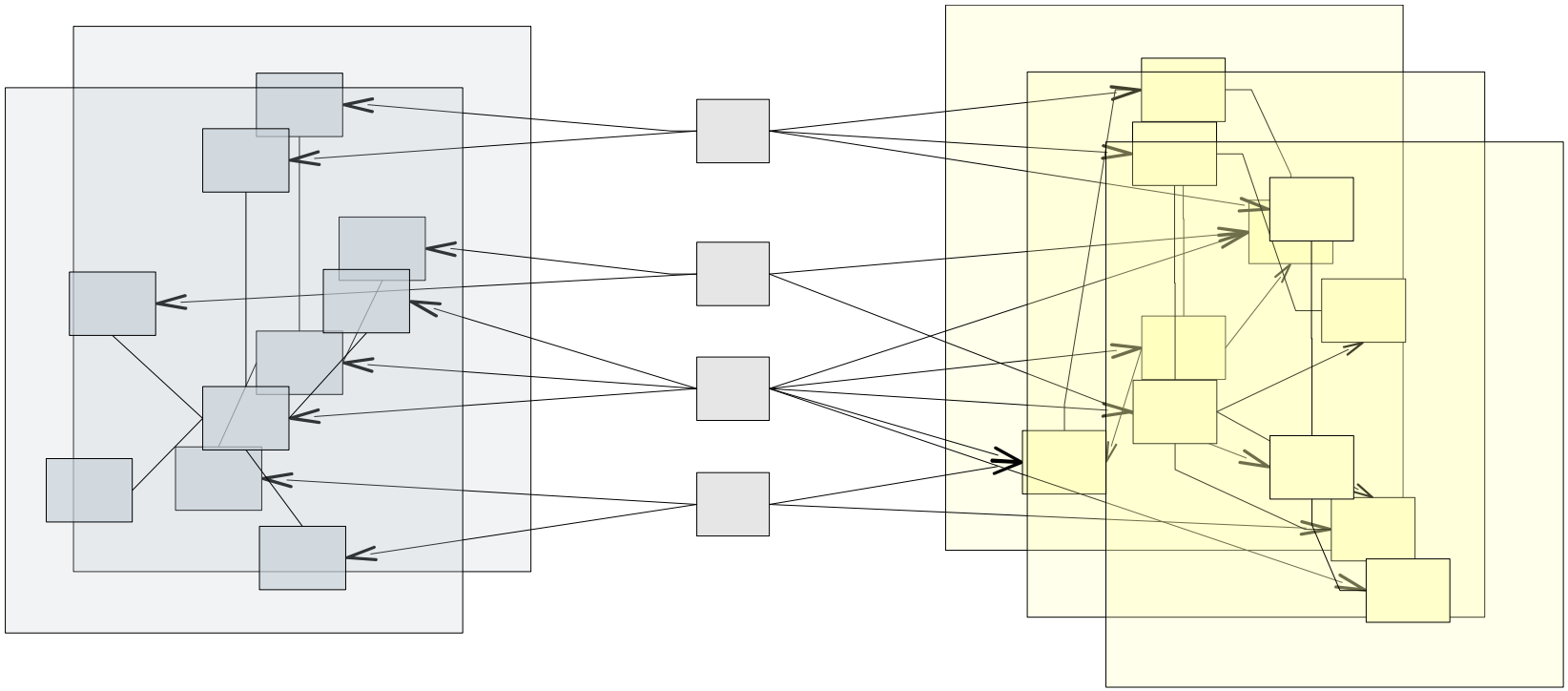


Starting a Transformation



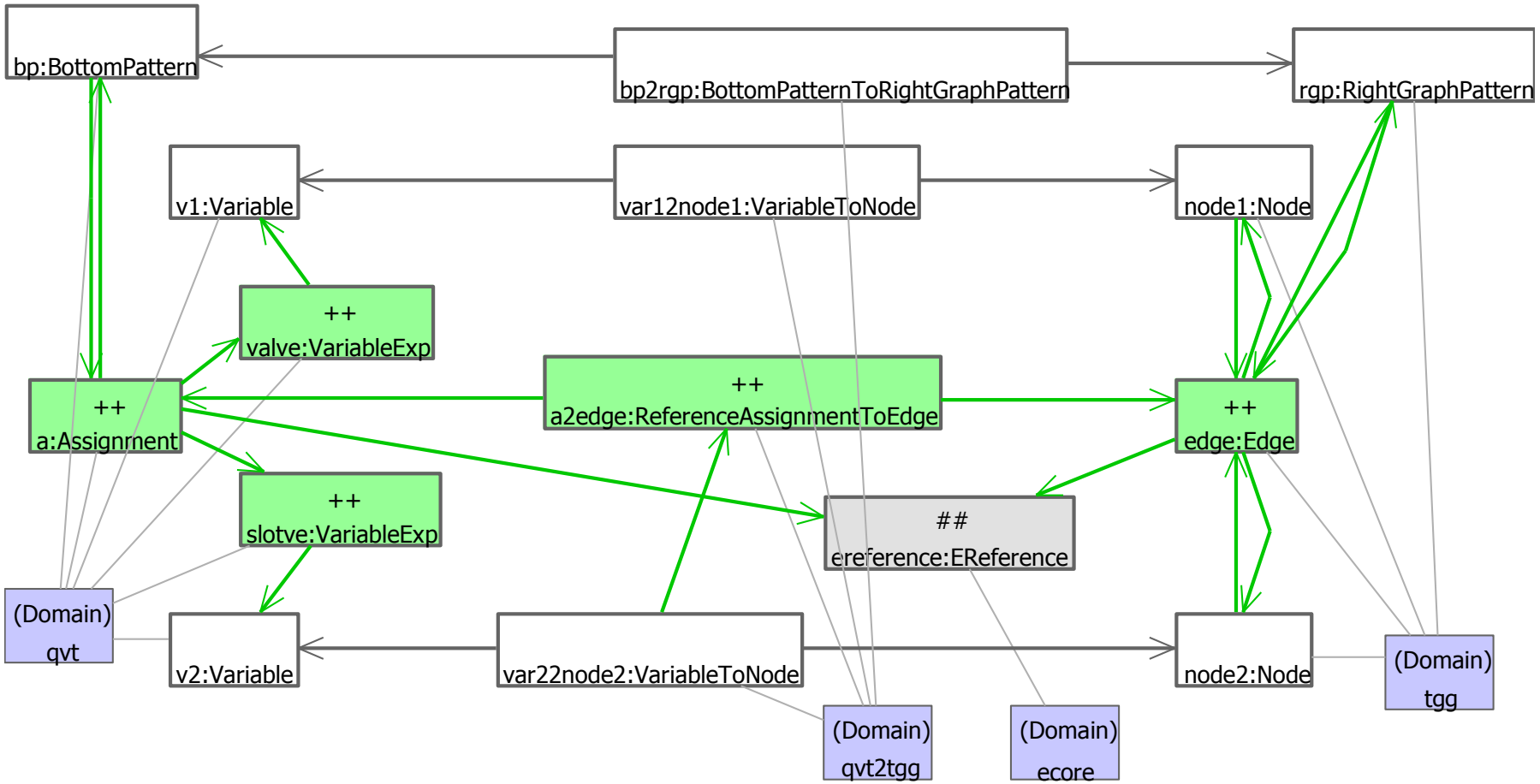
Further TGG-Interpreter Features

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



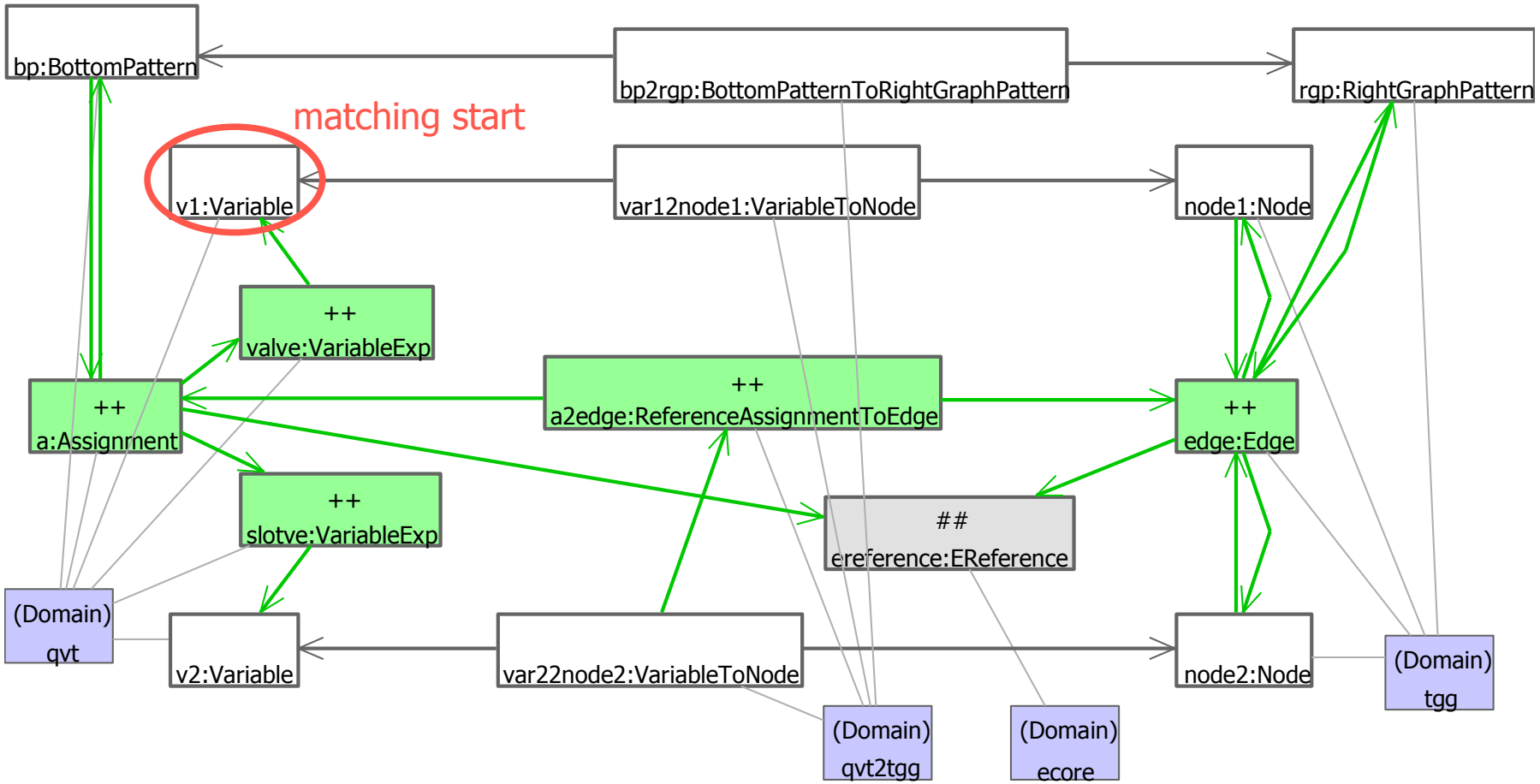
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



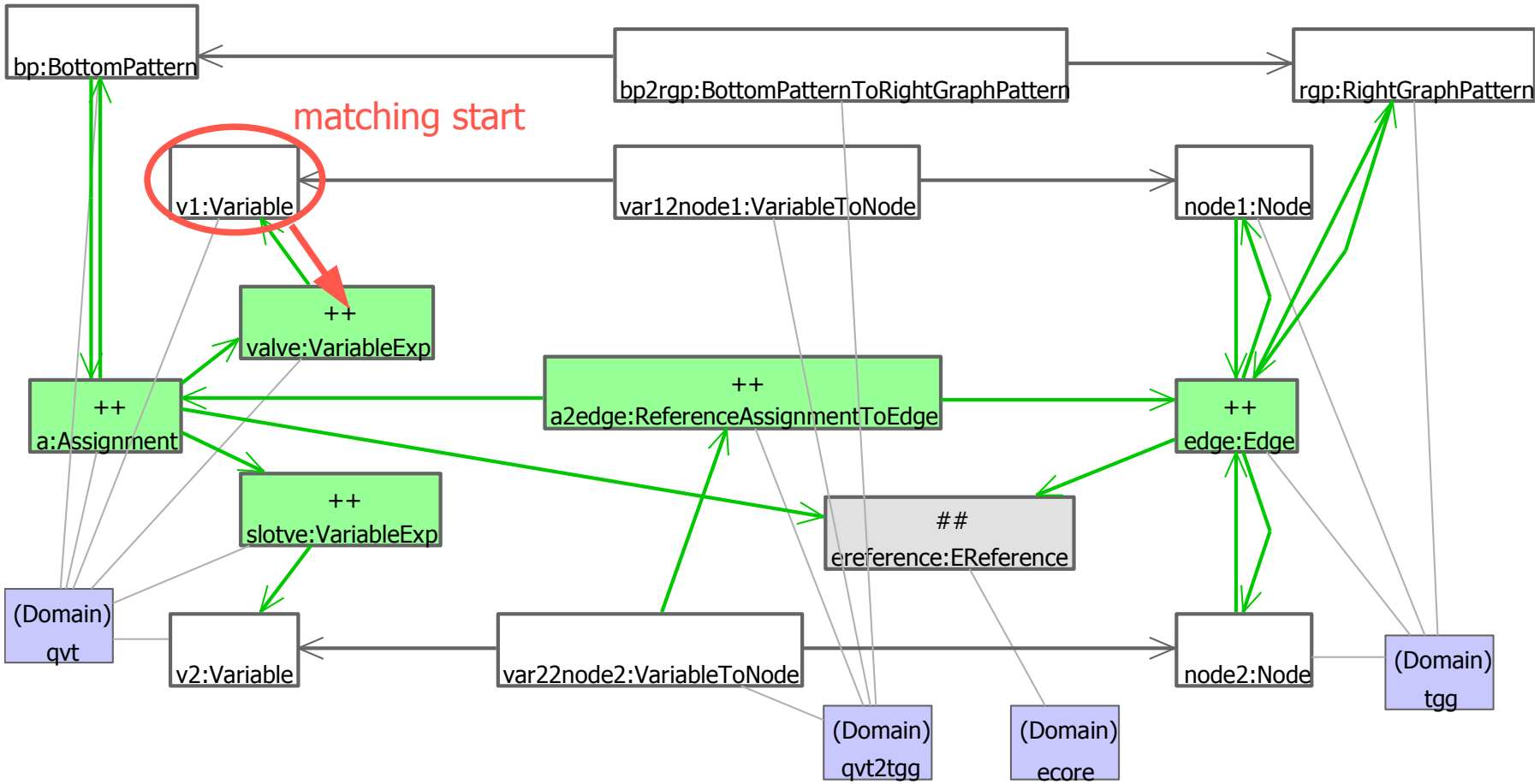
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



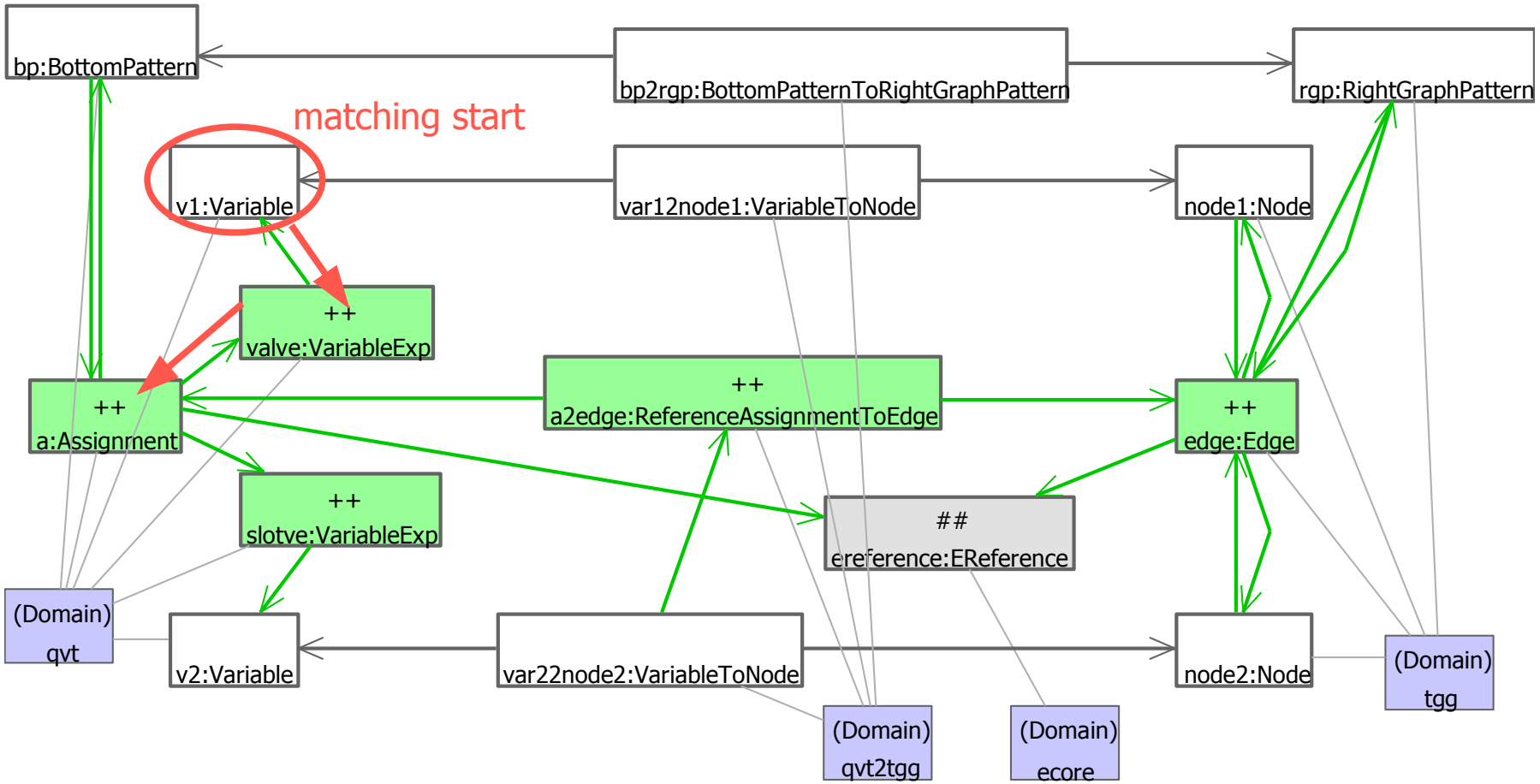
Further TGG-Interpreter Features

- No start node needed
- Navigating backwards on Edges



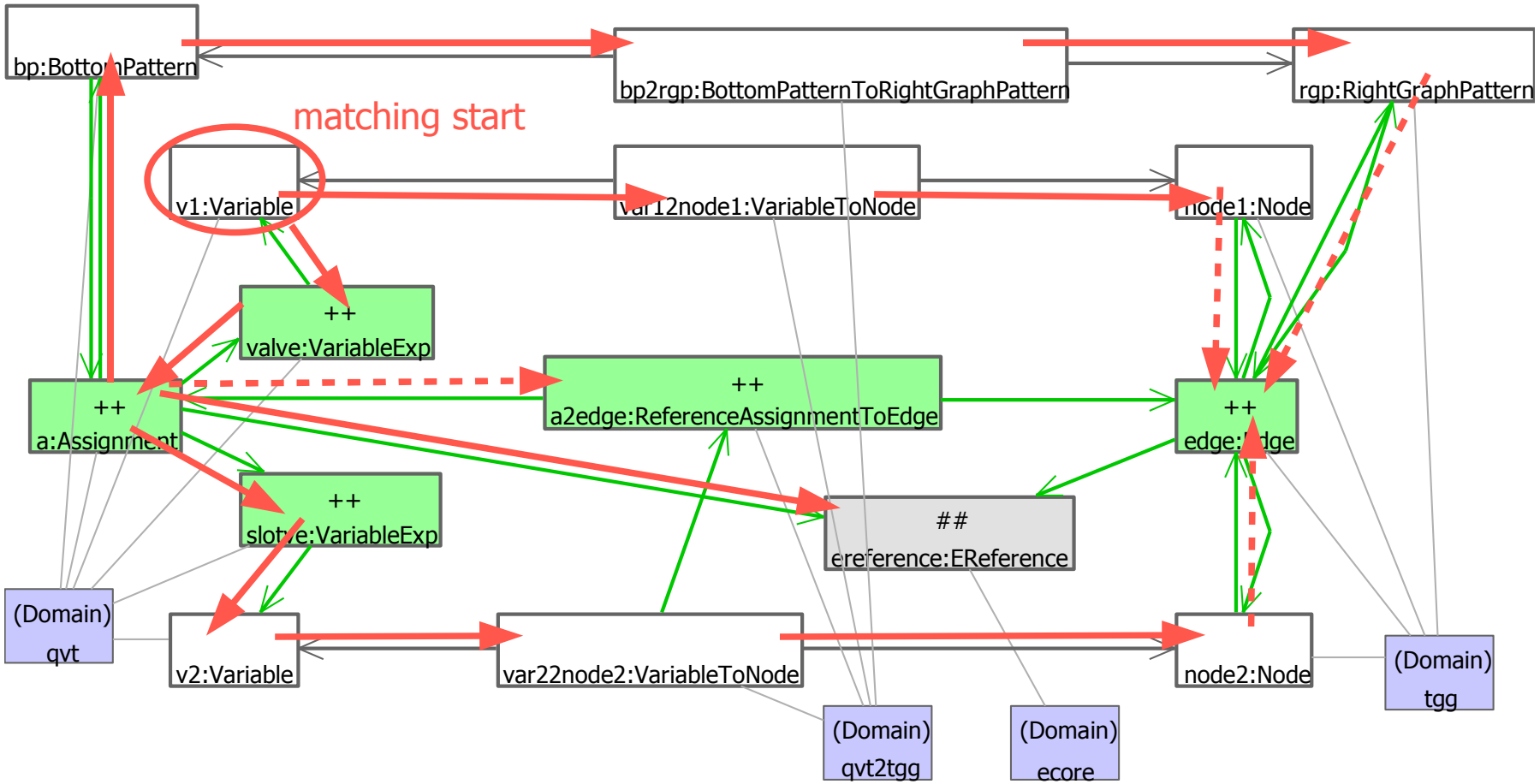
Further TGG-Interpreter Features

- 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 Conceptsin Computer Science,* Spinger 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