Type Annotation for Adaptive Systems

Paolo Bottoni, Andrew Fish, Francesco Parisi Presicce

Dipartimento di Informatica
Sapienza Università di Roma

University of Brighton
School of Computing, Engineering and Mathematics

Outline

- A view on type information
- Annotations on graphs
- From typing to type annotations
  - Correspondence patterns
  - Representing inheritance
- Dynamic typing
  - Case studies
- Conclusions

A view on type information

- Types constrain structure and behaviour of instances
- Inflexible when context changes
  - I have a book but I need a hammer
  - How do I derive that the book I have is “hammerisable”?
- Typical solutions
  - Merging information on domains
  - Creating bridges
- Rigidity connected with the notion of typing morphism

Types as annotations

- Annotations give information about properties
- Type morphisms substituted by annotation patterns
- Constraints associated with types
- Not instances of types but elements of a domain
  - Elements annotated with some type must conform to some pattern
  - Elements conforming to some pattern must be annotated with some type

Graphs with boxes (B-graphs)

$$G = (V, E, B, s, t, cnt)$$

- V, E as in usual graphs
- $s,t : E \to V \cup B$ source and target functions
- $cnt : B \to \mathcal{P}(V \cup B)$

Transitive containment, anti-symmetry not required, antireflexive

- $x \in cnt(b_1) \wedge b_1 \in cnt(b_2) \Rightarrow x \in cnt(b_2)$
- $b \notin cnt(b)$

Morphisms

A (B-)morphism f: $G_1 \to G_2$ *between B-graphs*

$G_i=(V_i\,, E_i\,, B_i\,, s_i\,, t_i\,, cnt_i)$ *is a triple*

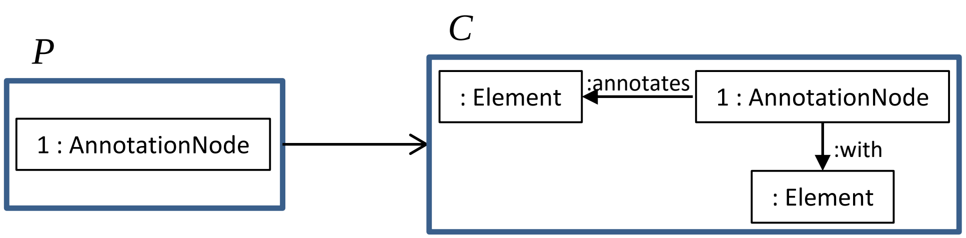
$(f_V\!: V_1 \to V_2\,,\ f_E\!: E_1 \to E_2\,, f_B\!: B_1 \to B_2)$ *that*

- *preserves* s_i *and* t_i*, and*
- *if* $x \in cnt(b_1)$ *then* $f_{V \cup B}(x) \in cnt(f_B(b_1))$

(composition of morphisms is constructed componentwise)

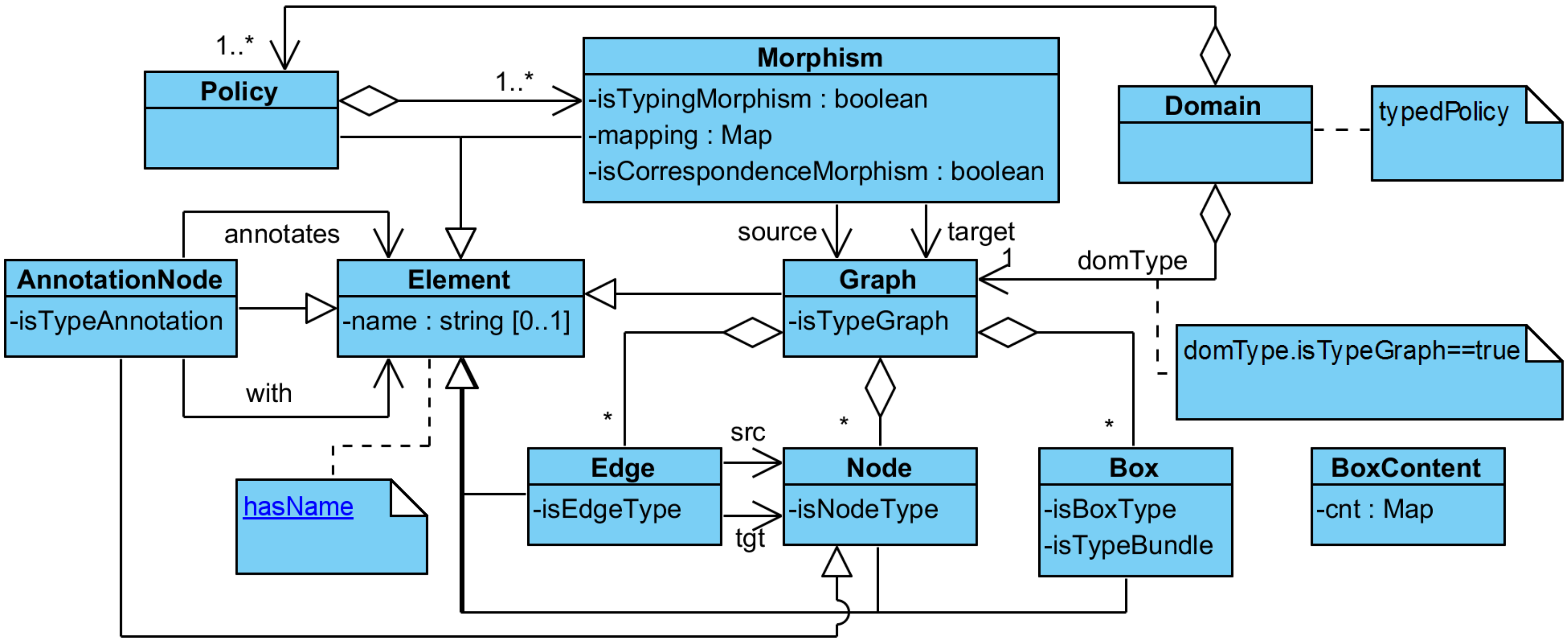
Annotations on graphs

- Annotations relate elements of two different domains via an annotation node.
- Constraints for well-formedness



Forbidden graphs

The metamodel *M* for type annotation



OCL constraints on *M*

context Domain **inv**

let

allMor : Set = Morphism.allInstances(),

type : Graph = self.graph

in

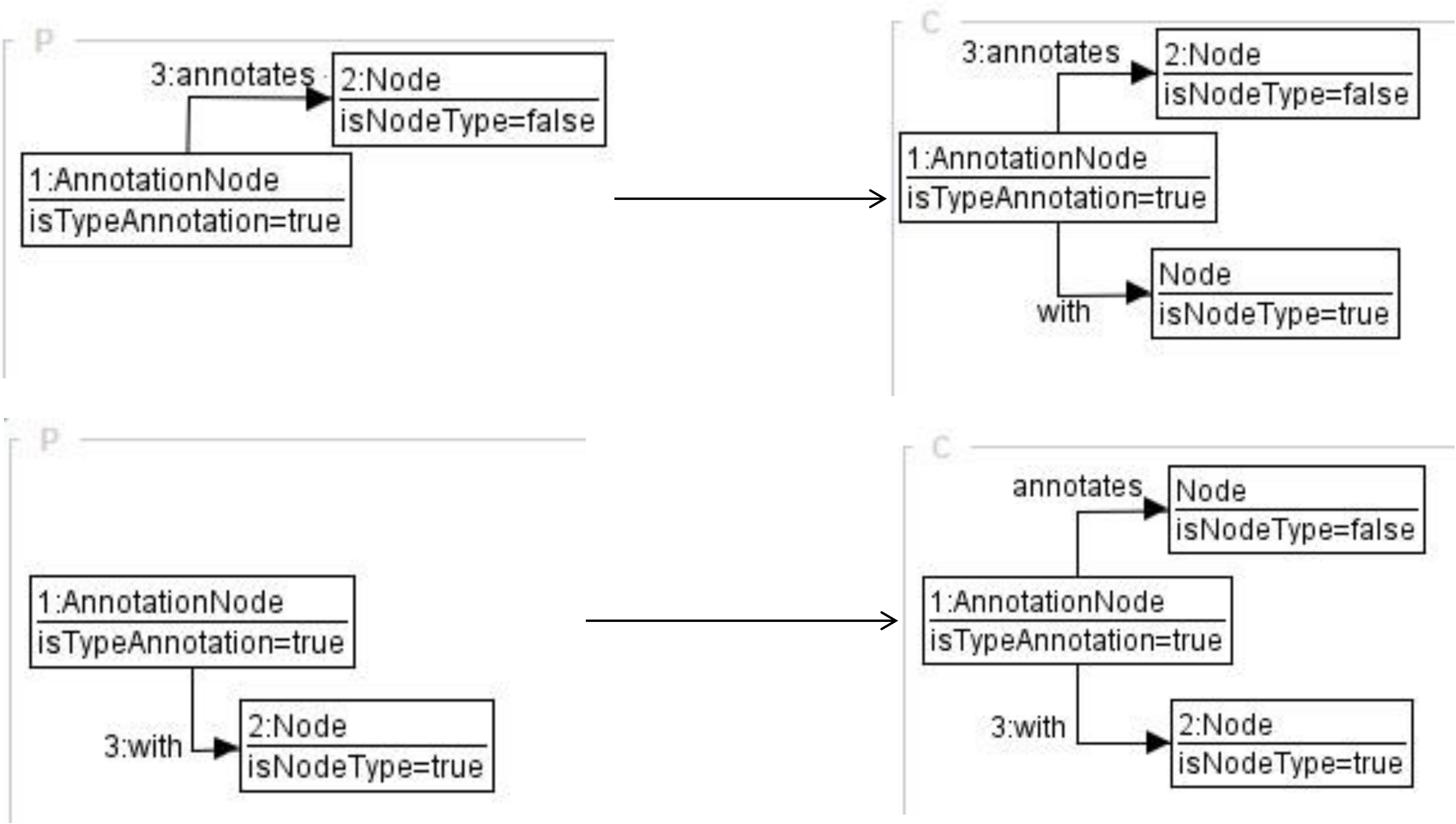
self.policy.morphism -> **forall** (m| allMor -> exists(m2,m3 |

m2.isTypingMorphism = true **and** m2.source = m.source **and** m2.target = self.domType **and**

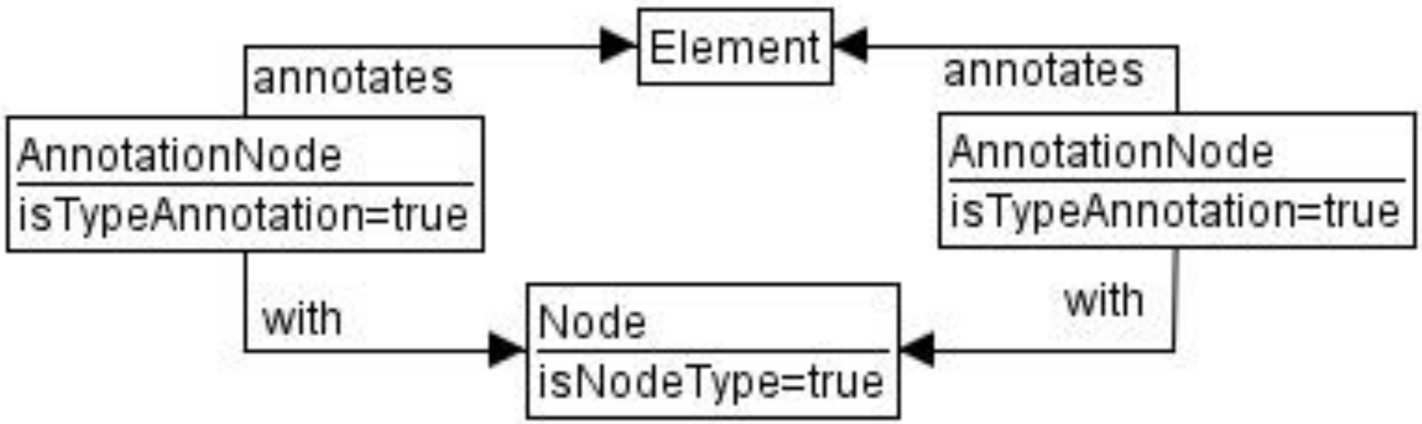
m3.isTypingMorphism = true **and** m3.source = m.target **and** m3.target = self.domType

))

Graph constraints for type annotation



Forbidden graph



For an element e,
$annType(e)$ is the set of types with which it is annotated

From typing to type annotations

$$\begin{tikzcd}
G \arrow[r, "in_1"'] \arrow[rr, bend left, "tp^g"] \arrow[dr, "f_g"'] & G\oplus TG \arrow[d, "f_g\oplus f_t"] & TG \arrow[l, "in_2"] \arrow[dl, "f_t"] \\
& H(G,tp^G) &
\end{tikzcd}$$

$H(G,tp^G)$, is the minimal graph with type annotation s.t.:

- both G and TG have isomorphic (disjoint) immersions in G, G' and TG', defined by morphisms f_g and f_t
- each element in G' is annotated with exactly one type element in TG';
- for each element x in G $annType(f_g(x)) = f_t(tp^G(x))$

The induced functor `typeAnn`: TG→AT1

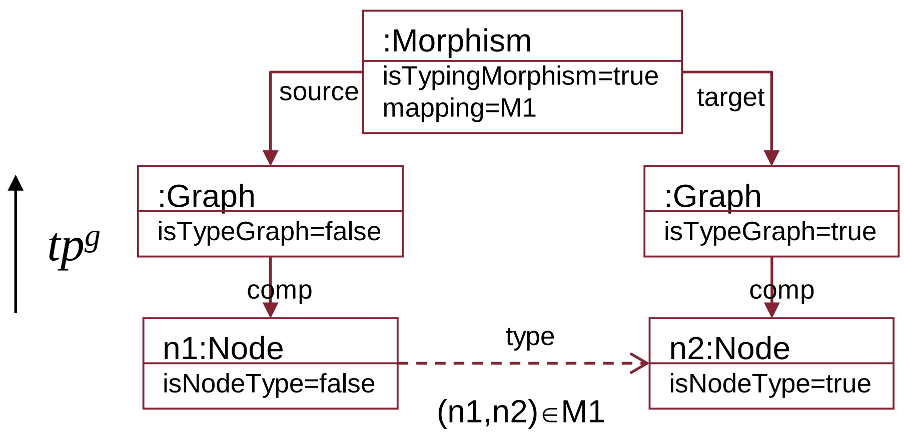
- $\texttt{typeAnn}_{\texttt{Ob}}$ maps each object G of **TG** into the object $H(G,tp^G)$ of **AT1**
- $\texttt{typeAnn}_{\texttt{Hom}}$ maps each morphism $m:G\rightarrow G'$ of **TG** into a morphism $m':H(G,tp^G)\rightarrow H(G',tp^{G'})$ such that for each element x of G $f_{g'}(m(x))= m'(f_g(x))$

Properties of `typeAnn`

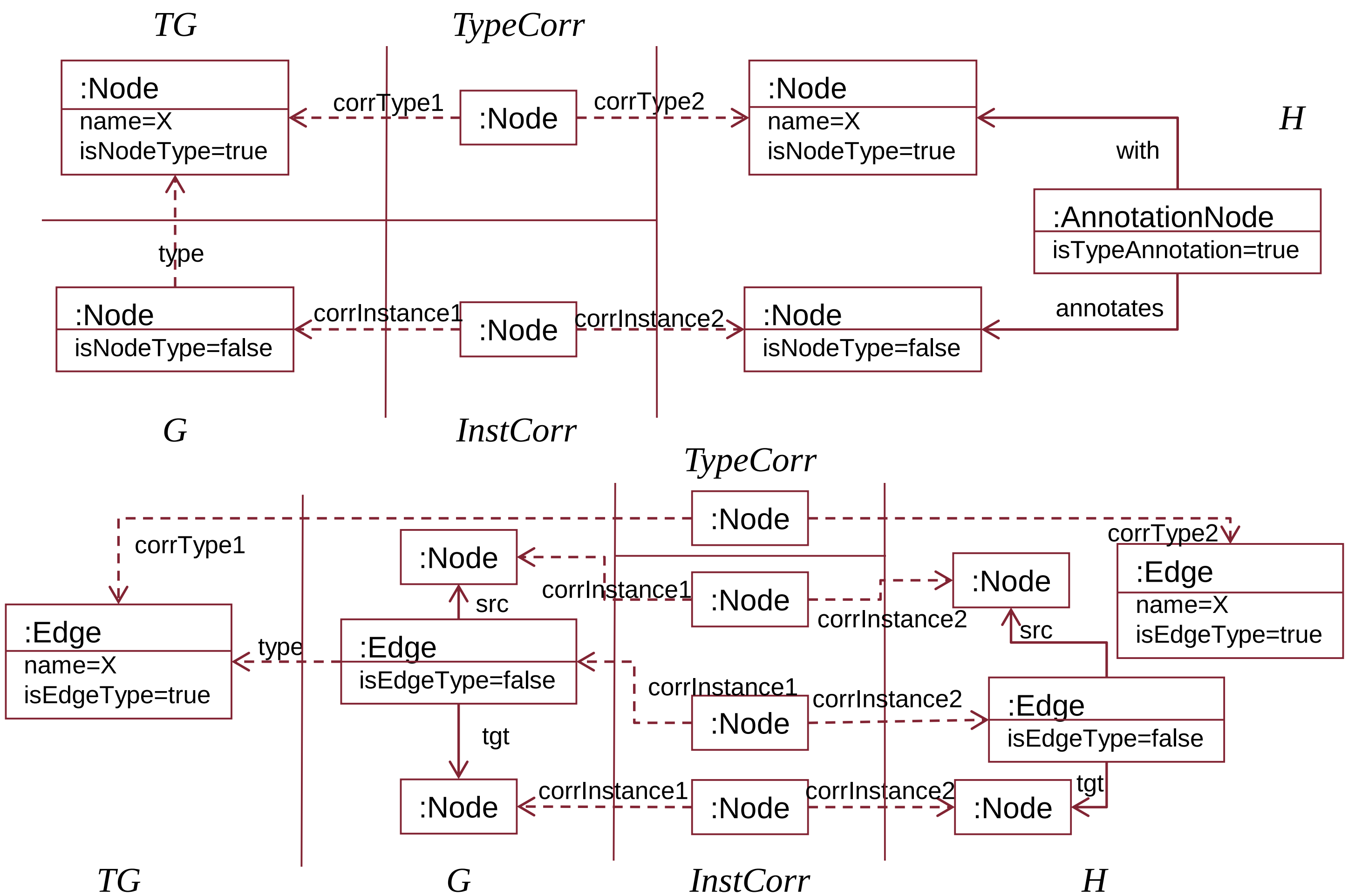
- **Lemma 1.** If a graph $G \in Ob(\mathbf{TG})$ is correct under typing morphisms, then its image under $\mathtt{typeAnn}_{\mathtt{Ob}}$ is correct under type annotation.
- **Lemma 2**. If a morphism $m \in Hom(\mathbf{TG})$ is type-preserving then $\mathtt{typeAnn}_{\mathtt{Hom}}(m)$ is type-annotation-preserving.

Correspondence patterns (in M) I

$$
\begin{tikzcd}
TG & TypeCorr \arrow[l, "corrType1"'] \arrow[dr, "corrType2"] & \\
 & & H(G,tp^G) \\
G \arrow[uu, "tp^g"'] & InstCorr \arrow[l, "corrInstance1"'] \arrow[ur, "corrInstance2"'] &
\end{tikzcd}
$$



Correspondence patterns (in *M*) II



Inheritance

$$T_1 \lhd \ldots \lhd T_i \lhd \ldots \lhd T_j \lhd \ldots \lhd \top$$

Typing

Type Annotation

el:T_i

el:Element $\xleftarrow{\text{annotates}}$:AnnotationNode isTypeAnnotation=true $\xrightarrow{\text{with}}$ b1:Box isTypeBundle=true

$$cnt(b1)=\{T_i,\ldots,T_j,\ldots\top\}$$

el1:T_i

?

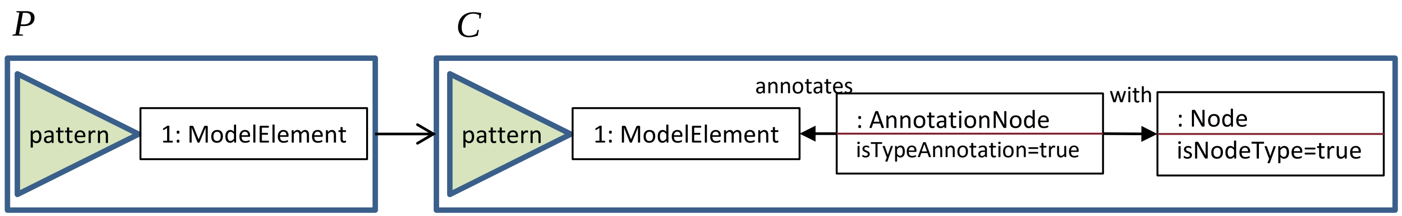
el2:T_j

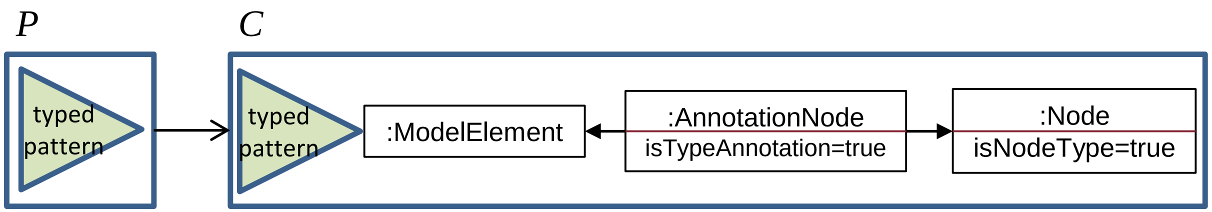
el:Element $\xleftarrow{\text{annotates}}$:AnnotationNode isTypeAnnotation=true $\xrightarrow{\text{with}}$ b2:Box isTypeBundle=true

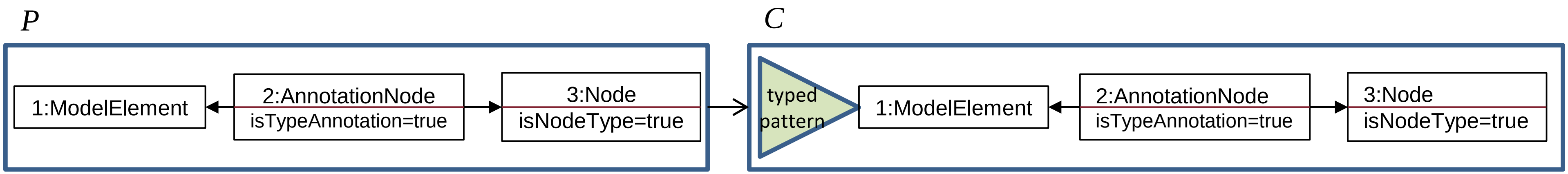
$$cnt(b2)=\{T_j,\ldots\top\}$$

Instance Level Type Change

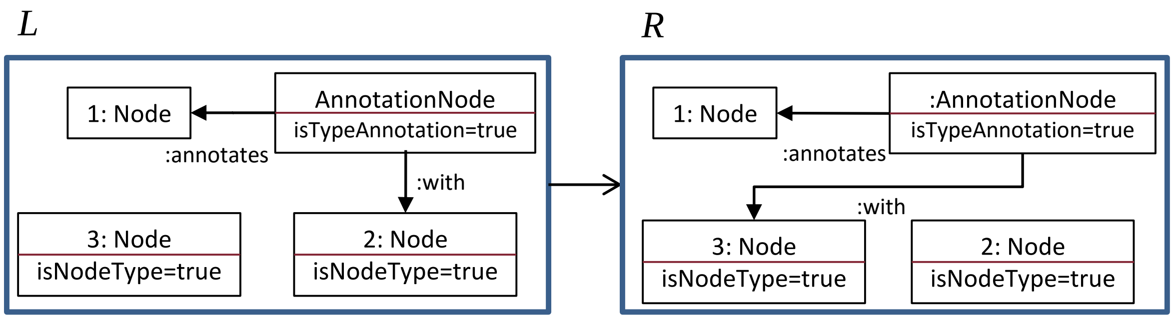
Patterns and types







Type change and repair actions

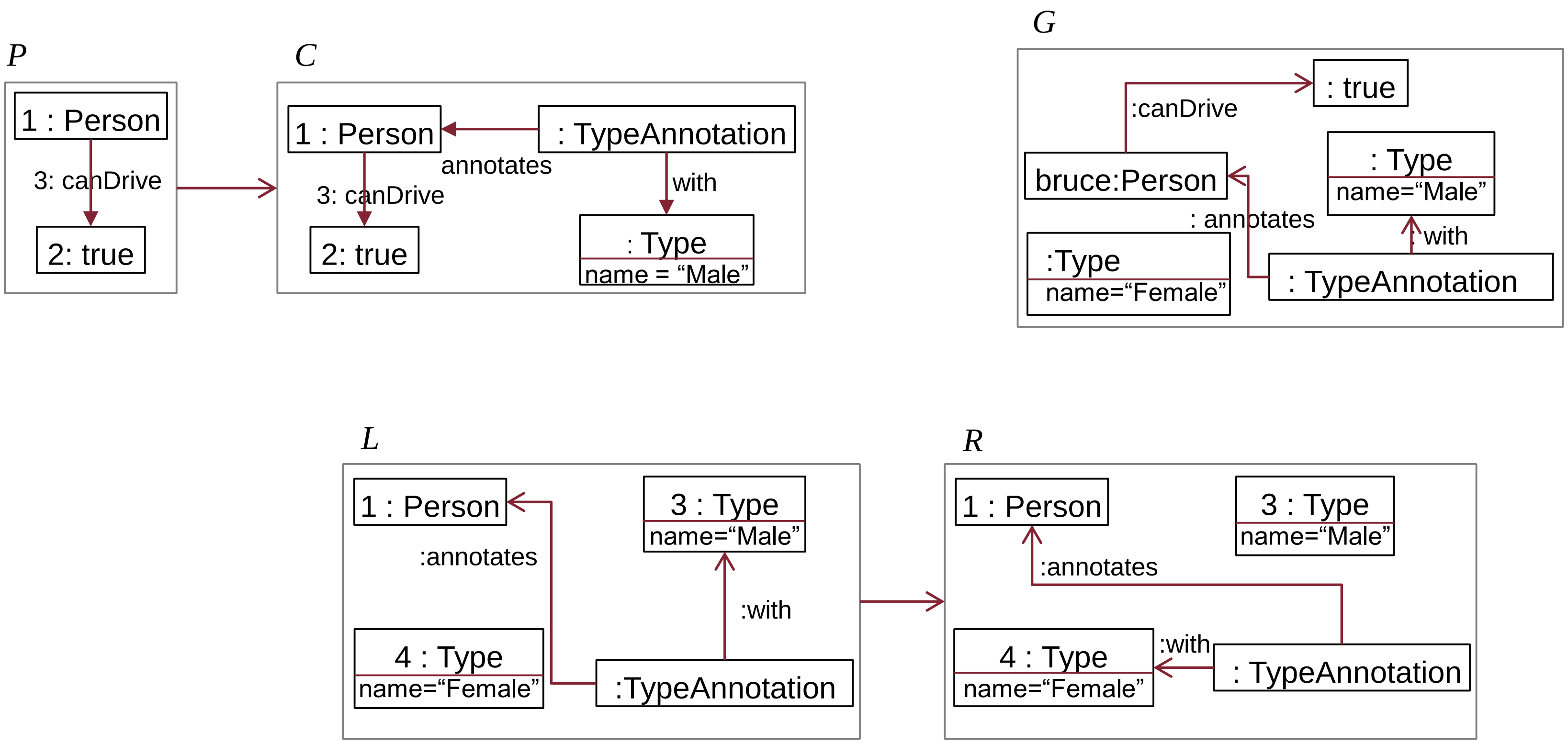


$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\
G & D \arrow[l] \arrow[r] & H \\
C \arrow[u] & P \arrow[l] \arrow[r] \arrow[u] & C \arrow[u, dashed]
\end{tikzcd}$$

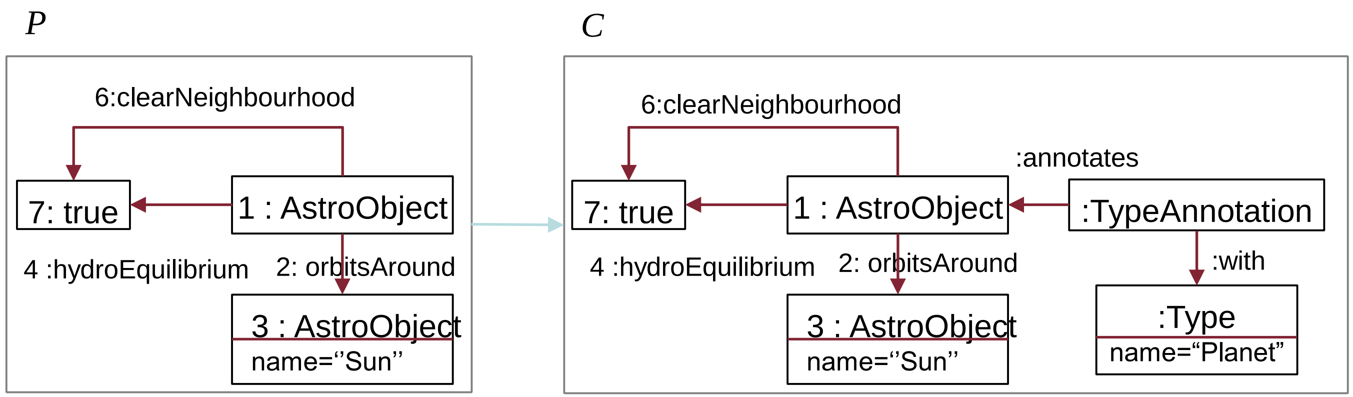
$$\begin{tikzcd}
L \oplus_e P \arrow[d] & K \oplus_e \overline{P} \arrow[l] \arrow[r] \arrow[d] & R \oplus_e \overline{P} \arrow[d] \\
G & D \arrow[l] \arrow[r] & H \\
C \arrow[u] & P \arrow[l] \arrow[ul] \arrow[u, dotted, "x"] &
\end{tikzcd}$$

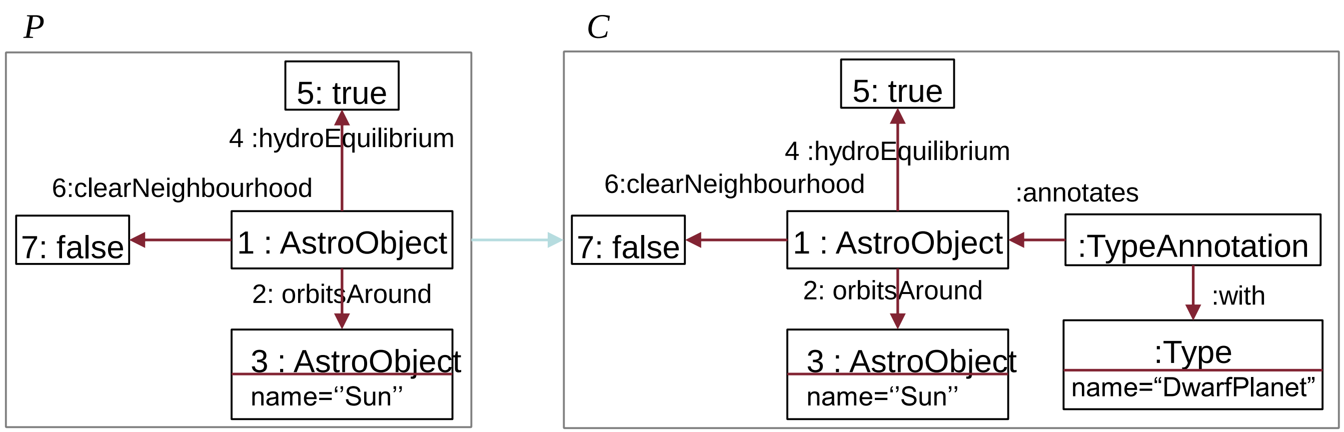
$$\begin{tikzcd}
L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] & R \arrow[dr] & & P \arrow[dl] & \overline{P} \arrow[l] \arrow[r] \arrow[d] & \overline{P} \arrow[d] \\
G & D \arrow[l] \arrow[rr] & & H & & D' \arrow[ll] \arrow[r] & H' \\
C \arrow[u] & P \arrow[l] \arrow[r] \arrow[u] & C \arrow[ur, dotted, "x"] & & C \arrow[ul, dotted, "x"] & P \arrow[l] \arrow[u, dotted, "x"] &
\end{tikzcd}$$

Case study: gender change

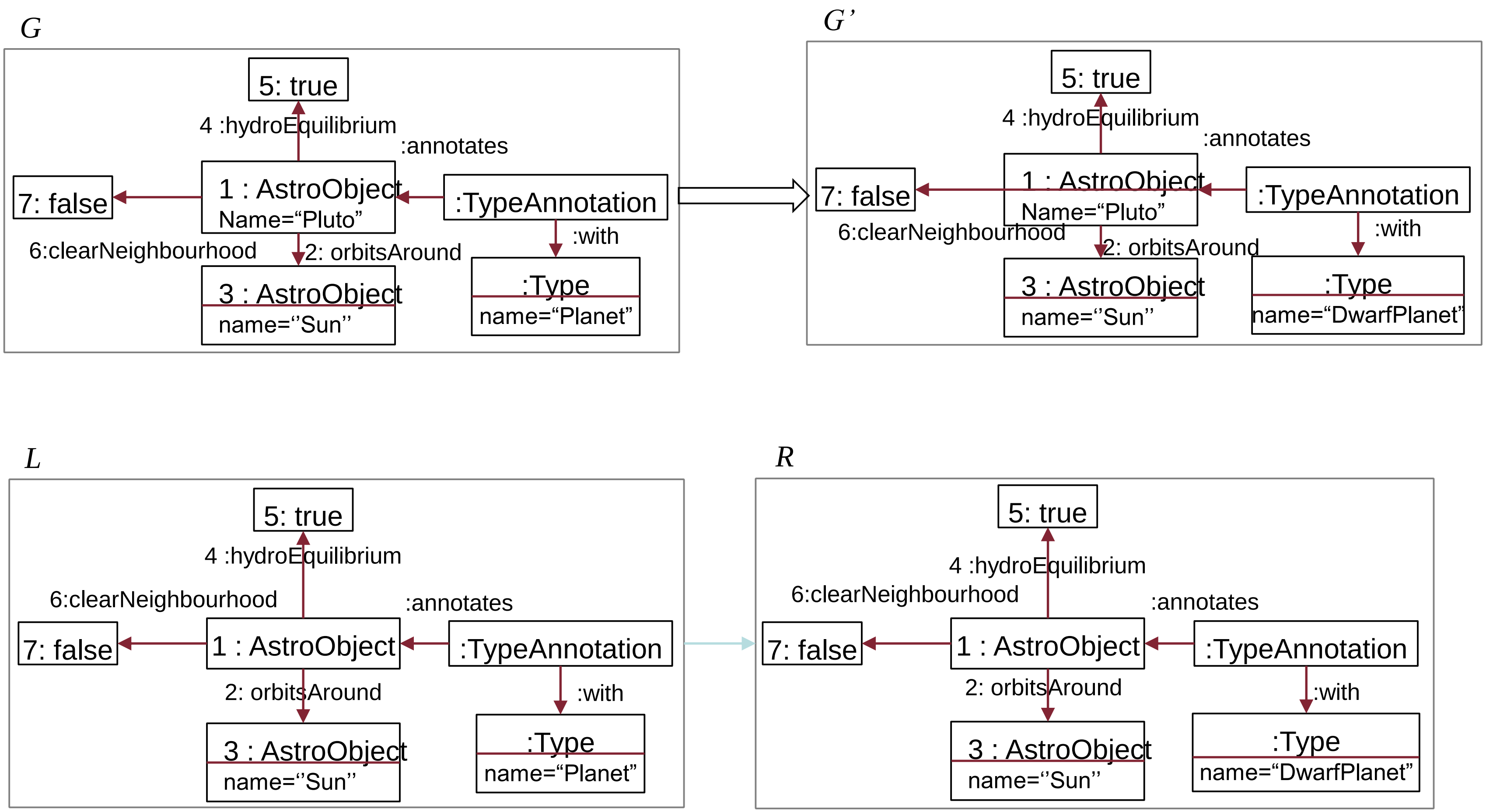


Case study: classification





Case study: reclassification



Conclusions

- Type annotations as a flexible way to associate type information to instances
- Types integrated in the domain graphs
- Consequences:
  - Dynamic typing possible at the instance level
  - Multiple typing inherently supported
  - Additional information can be associated with elements, even if not required by their types