Type Annotation for Adaptive Systems

Paolo Bottoni, Andrew Fish, Francesco Parisi Presicce
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)

\[ G = (V, E, B, s, t, \text{cnt}) \]

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

Transitive containment, anti-symmetry not required, antireflexive

- \( x \in \text{cnt}(b_1) \land b_1 \in \text{cnt}(b_2) \Rightarrow x \in \text{cnt}(b_2) \)
- \( b \not\in \text{cnt}(b) \)
Morphisms

A (B-)morphism $f: G_1 \rightarrow G_2$ between B-graphs $G_i = (V_i, E_i, B_i, s_i, t_i, \text{cnt}_i)$ is a triple $(f_V: V_1 \rightarrow V_2, f_E: E_1 \rightarrow E_2, f_B: B_1 \rightarrow B_2)$ that
- preserves $s_i$ and $t_i$, and
- if $x \in \text{cnt}(b_1)$ then $f_{V \cup B}(x) \in \text{cnt}(f_B(b_1))$

(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**

\[
\text{allMor} : \text{Set} = \text{Morphism.allInstances}(), \\
\text{type} : \text{Graph} = \text{self.graph}
\]

**in**

\[
\text{self.policy.morphism} \rightarrow \text{forall} (m \mid \text{allMor} \rightarrow \exists (m2, m3 \mid \\
\quad m2.\text{isTypingMorphism} = \text{true and } m2.\text{source} = m.\text{source and } m2.\text{target} = \\
\quad \text{self.domType and } m3.\text{isTypingMorphism} = \text{true and } m3.\text{source} = m.\text{target and } m3.\text{target} = \\
\quad \text{self.domType})
\]

Eindhoven, April 3, 206

GaM 2016
Graph constraints for type annotation

[Diagram showing relationships between nodes and annotations with properties such as `isNodeType` and `isTypeAnnotation`.]
Forbidden graph

For an element \( e \),
\( \text{annType}(e) \) is the set of types with which it is annotated
From typing to type annotations

\[ H(G, tp^G) \], 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 \( G \)  \( \text{annType}(f_g(x)) = f_t(tp^G(x)) \)
The induced functor $\text{typeAnn}: \mathcal{T}G \to \mathcal{AT}_1$

- $\text{typeAnn}_{\text{Ob}}$ maps each object $G$ of $\mathcal{T}G$ into the object $H(G, tp^G)$ of $\mathcal{AT}_1$

- $\text{typeAnn}_{\text{Hom}}$ maps each morphism $m: G \to G'$ of $\mathcal{T}G$ into a morphism $m': H(G, tp^G) \to H(G', tp^{G'})$ such that for each element $x$ of $G$, $f_{g} \cdot (m(x)) = m'(f_{g}(x))$
Properties of typeAnn

• **Lemma 1.** If a graph \( G \in \text{Ob}(\mathcal{T}G) \) is correct under typing morphisms, then its image under \( \text{typeAnn}_{\text{Ob}} \) is correct under type annotation.

• **Lemma 2.** If a morphism \( m \in \text{Hom}(\mathcal{T}G) \) is type-preserving then \( \text{typeAnn}_{\text{Hom}}(m) \) is type-annotation-preserving.
Correspondence patterns (in $M$) I

$TG \xleftarrow{corrType1} TypeCorr \xrightarrow{corrType2} H(G,tp^G)$

$G \xleftarrow{corrInstance1} InstCorr \xrightarrow{corrInstance2}$

$\langle n_1, n_2 \rangle \in M_1$

$\langle G, tp^g \rangle$

$\langle G, tp^g \rangle$

$\langle G, tp^g \rangle$

$\langle G, tp^g \rangle$

$\langle G, tp^g \rangle$
Correspondence patterns (in $M$) II
Inheritance

\[ T_1 \circ \ldots \circ T_i \circ \ldots \circ T_j \circ \ldots \circ T \]

**Typing**

\[ \text{el:T}_i \]

**Type Annotation**

\[ \text{el:Element} \xrightarrow{\text{annotates}} \text{:AnnotationNode} \]

\[ \text{isTypeAnnotation}=\text{true} \]

\[ \text{with} \]

\[ \text{b1:Box} \]

\[ \text{isTypeBundle}=\text{true} \]

\[ \text{cnt}(b1)=\{ T_i, \ldots, T_j, \ldots, T \} \]

**Instance Level Type Change**

\[ \text{el1:T}_i \]

\[ \text{el2:T}_j \]

\[ \text{el:Element} \xrightarrow{\text{annotates}} \text{:AnnotationNode} \]

\[ \text{isTypeAnnotation}=\text{true} \]

\[ \text{with} \]

\[ \text{b2:Box} \]

\[ \text{isTypeBundle}=\text{true} \]

\[ \text{cnt}(b2)=\{ T_j, \ldots, T \} \]
Type change and repair actions

$L$:
- 1: Node
  - :isTypeAnnotation=true
- 2: Node
  - :isNodeType=true
- 3: Node
  - :isNodeType=true

$R$:
- 1: Node
  - :isTypeAnnotation=true
- 2: Node
  - :isNodeType=true
- 3: Node
  - :isNodeType=true

$L \leftarrow K \rightarrow R$

$L \oplus_c P \leftarrow K \oplus_c P \rightarrow R \oplus_c P$

$G \leftarrow D \rightarrow H$

$C \leftarrow P \rightarrow C$
Case study: gender change

Eindhoven, April 3, 206
GaM 2016

20
Case study: classification

$P$

$C$

$P$

$C$

Eindhoven, April 3, 206  GaM 2016
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