Bidirectionalizing ATL within Echo

Nuno Macedo Alcino Cunha

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY
INESCTEC

Universidade do Minho

FATBIT Workshop 2013
October 3, Braga, Portugal

ATL

- *ATL* is a widely used model transformation language;
- Has a *de facto* operational semantics implemented as an Eclipse plugin (denoted by $\overrightarrow{\mathbf{t}}$);
- However, it is *unidirectional*
    - Specification only states how to create a model N from a model M.

ATL and Echo

- The prescribed method to achieve bidirectionality is to define two transformations
  - Correctness and maintainability problem;
- ATL was created to answer the original QVT RFP
  - It shares some characteristics with the QVT standard;
- We propose the bidirectionalization of (the *declarative* subset of) ATL transformations with *Echo*.

ATL Language

- ATL main constituents are *rules*:

$$
\begin{array}{l}
[[\textbf{unique}]\ \textbf{lazy}]\ \textbf{rule}\ R\ \{ \\
\quad \textbf{from}\ a : A\ (\pi_M) \\
\quad \textbf{to}\ b : B\ (\phi) \\
\}
\end{array}
$$

- Induce implicit traceability links.
- Bindings may rely on implicit rule calls.

Example


```
module employ;
create OUT : Company from IN : World;
//PersonToEmployee
rule P2E {
  from
    p : World!Person ()
  to
    e : Company!Employee (
      name <- p.name
    )
}
```

ATL Semantics

- Disregards existing target model;
- *Matched rules*:
    - For every source a that matched π_M *create* a b with bindings ϕ
- *Unique laze rules:*
    - Only run if called by other rules.

Bidirectional framework

- Lenses ($\overrightarrow{\mathbf{t}}(\overleftarrow{\mathbf{T}}(m, n)) = n$ and $\overleftarrow{\mathbf{T}}(m, \overrightarrow{\mathbf{t}}(m)) = m$)?
  - Infer relation such that $\mathbf{T}(m, n) \equiv n = \overrightarrow{\mathbf{t}}(m)$;
  - Would fail for *non-surjective* transformations;
  - Could relax the laws (to those of GRoundTram);
  - Applying $\overrightarrow{\mathbf{t}}$ would still disregard updates on n.
- Maintainers?
  - Both $\overleftarrow{\mathbf{T}}$ and $\overrightarrow{\mathbf{T}}$ derived from the inferred $\mathbf{T}$;
  - $\overrightarrow{\mathbf{t}}$ would be run once to generate the initial target model..
  - ... and $\overleftarrow{\mathbf{T}}$ and $\overrightarrow{\mathbf{T}}$ would maintain them consistent;
  - Behavior of $\overrightarrow{\mathbf{t}}$ and $\overrightarrow{\mathbf{T}}$ converge for surjective transformations.

Consistency relation

- Source patterns are domain patterns (OCL predicates);
- Bindings are `where` constraints (post-conditions);
- *Forall-there-exists* tests do not suffice:
    - there must exist *exactly one* match;
- Requires traceability links.

Example

- E.g., P2E:

$$
\begin{array}{l}
\mathtt{P2E}_{\blacktriangleright}\ (in : \mathtt{World}, out : \mathtt{Company}) \equiv \\
\quad \exists\ \mathtt{P2E}_{\triangleleft\triangleright} \subseteq \mathtt{Person} \times \mathtt{Employee} \mid \\
\qquad \forall\ p : \mathtt{Person} \mid \exists !\ e : \mathtt{Employee} \mid \\
\qquad\quad \mathtt{P2E}_{\triangleleft\triangleright}\ (p, e) \wedge p.\mathtt{name} = e.\mathtt{name} \quad \wedge \\
\qquad \forall\ e : \mathtt{Employee} \mid \exists !\ p : \mathtt{Person} \mid \\
\qquad\quad \mathtt{P2E}_{\triangleleft\triangleright}\ (p, e) \wedge p.\mathtt{name} = e.\mathtt{name}
\end{array}
$$

Unique lazy rules

- Are one-to-one but only if called;
- Existence is relaxed from the definition...

$$R_{\blacktriangleright}\ (m : M, n : N) \equiv \exists\ R_{\lhd\rhd} \subseteq A \times B \mid \\ \forall\ a : A, b : B \mid R_{\lhd\rhd}\ (a, b) \wedge \pi_M \Rightarrow \phi$$

- ... and is rather enforced by the rule call

$$\exists\,!\ b : B \mid R_{\lhd\rhd}\ (e, b) \wedge \forall\ a : A \mid R_{\lhd\rhd}\ (a, b) \Rightarrow a = e$$

Implicit calls

- Our implementation also relies on traceability links $R_{\triangleleft\triangleright}$;
- ATL states the domain of rules must be disjoint;
- Matched target element can be retrieved.

Echo

- Higher-order existential quantifications are implemented in Alloy through skolemization;
- ATL is already deployed as an Eclipse plugin:
  - Like Echo, it is built over EMF;
- An embedding of the technique in Echo is under way.