A Domain-Specific Language with Semantics for Shot Puzzles

COREF work in progress

Presented at *SET Meeting*

11 January 2011, TU/e

Tom Verhoeff

In cooperation with *Ulyana Tikhonova* and *Maarten Manders*

Eindhoven University of Technology

Dept. of Math. & CS

Model-Drive Software Engineering

Software Engineering & Technology

Motivation behind COREF

- Software components are systems exhibiting behavior
- Wanted: tools to define, to generate code, to analyze, to . . .
- Approach: Domain Specific Language (syntax) to describe . . .
- Define formal semantics for this language *once* in DSL framework
- Use model transformations to realize tools

Goals of Shot Puzzle Case Study

- Become familiar with state of the art in DSL tooling

  Use Eclipse Modeling Framework (EMF)

- Find out what it takes to define semantics in a DSL framework

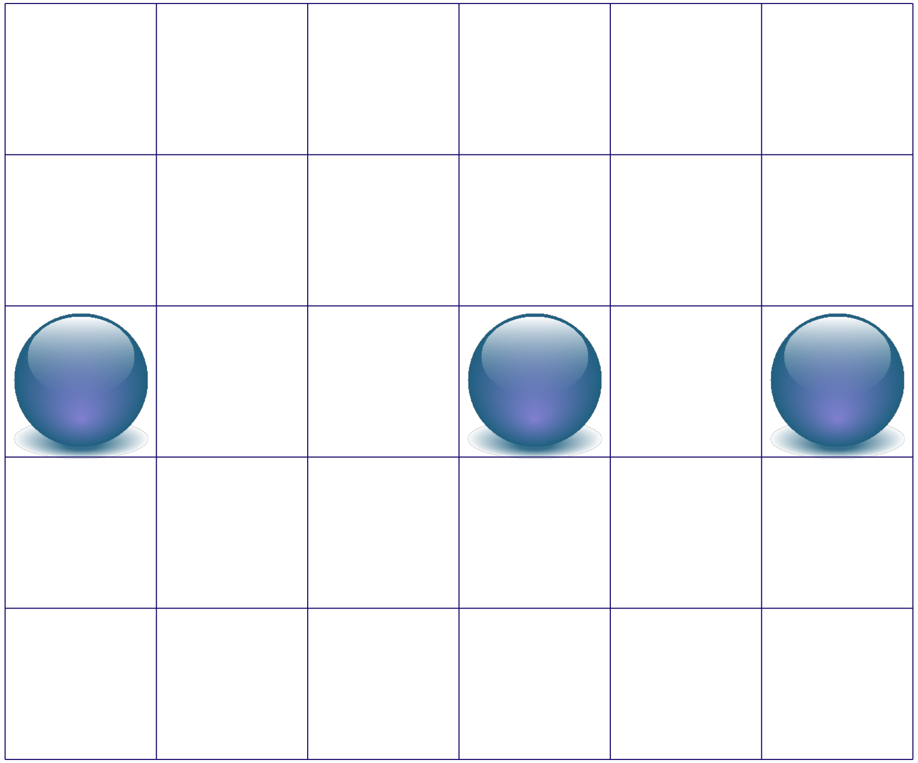
  Take ad hoc approach

Why Shot puzzles:

- involve behavior

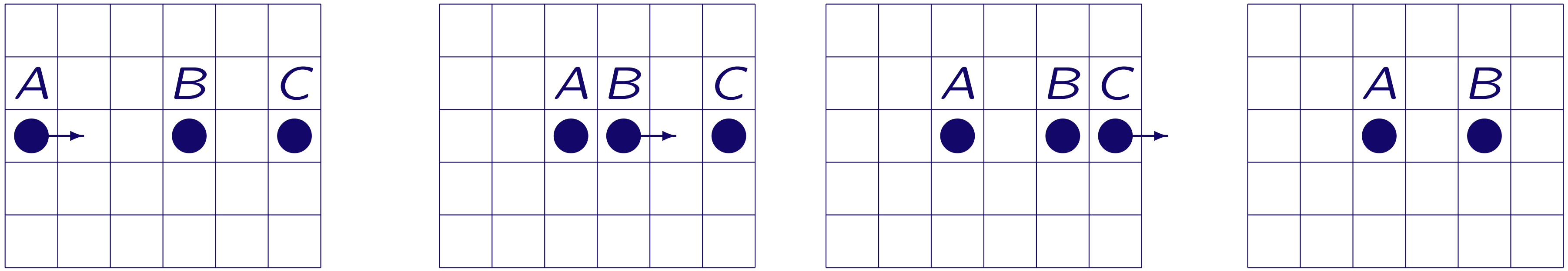
- are simple but not trivial

Shot Puzzle



Shot Puzzle: Dynamics (informal)

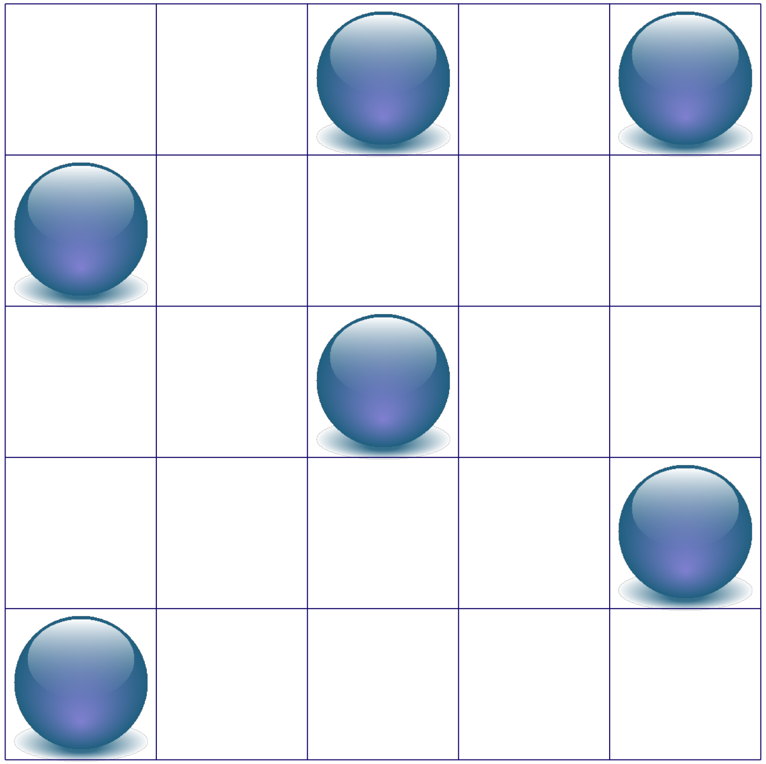
- A *move* consists of pushing a marble (horizontally or vertically) over an empty cell hitting another marble.
- The push propagates across a chain of aligned marbles.
- The last marble being hit disappears from the grid.



start configuration   intermediate 1   intermediate 2   end configuration

Shot Puzzle: Objective

The objective is to remove all marbles but one.



Mathematical Model for Shot Puzzles: Statics

A **Shot puzzle** is a triple (M, G, g), where

- M is a finite set (the marbles),
- $G \subseteq \mathbb{Z} \times \mathbb{Z}$ (the grid), and
- $g : M \rightarrowtail G$ is an injection from M to G (marble locations).

Mapping g is called the **initial configuration** of the puzzle.

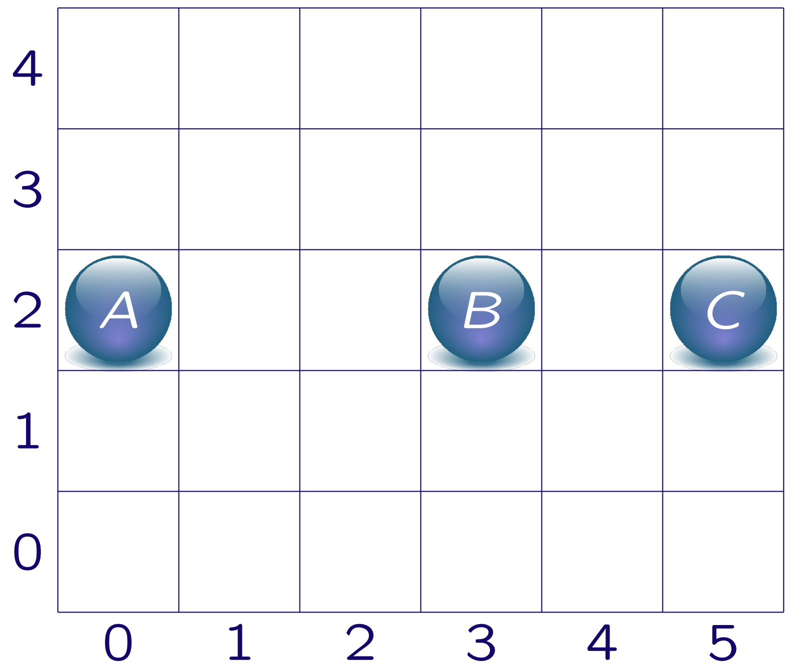
Injection: no two marbles at the same location

Mathematical Model for Shot Puzzles: Example

$$(\{A, B, C\}, G, \{A \mapsto (0,2), B \mapsto (3,2), C \mapsto (5,2)\})$$

where

$$G = \mathbb{Z}_6 \times \mathbb{Z}_5 = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid 0 \leq x < 6 \ \wedge \ 0 \leq y < 5\}$$



Mathematical Model for Shot Puzzles: Push

A push is a pair (m, v), where

- $m \in M$ is a marble of the puzzle, and
- $v \in DIR = \{ EAST, NORTH, WEST, SOUTH \}$ is a direction vector with

$$\begin{aligned} EAST &= (1, 0) \\ NORTH &= (0, 1) \\ WEST &= (-1, 0) \\ SOUTH &= (0, -1) \end{aligned}$$

Mathematical Model for Shot Puzzles: Current Configuration

The current configuration is given by a *partial* injection from M to G, that is, $g : M \rightarrowtail\mapsto G$.

The set of marbles present on the grid is given by $\mathrm{dom}(g)$, that is, the subset of M for which g is defined.

The range of g, denoted by $\mathrm{ran}(g)$, is the set of grid locations with a marble.

Mathematical Model for Shot Puzzles: Push Applicability

A push (m, v) can be applied in configuration g whenever

- m is present on the grid: $m \in \mathrm{dom}(g)$, and
- its v-neighbor is empty: $g(m) + v \notin \mathrm{ran}(g)$, and
- there exists a $k \in \mathbb{N}^+$ such that
  - m 'sees' a marble in direction v: $g(m) + kv \in \mathrm{ran}(g)$, and
  - the path to that marble belongs to the grid:

$$\{\, \ell : \mathbb{N} \mid 1 \leq \ell \leq k \bullet g(m) + \ell v \,\} \subseteq G$$

The latter condition is superfluous when the grid is convex.

Mathematical Model for Shot Puzzles: Dynamic Configuration

A dynamic configuration is a pair (g, h) of a configuration and a push mapping $h : M \nrightarrow DIR$, such that

- only marbles present on the grid can have a push:

  $$\mathrm{dom}(h) \subseteq \mathrm{dom}(g)$$

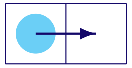
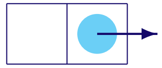
- and at most one marble has a push:

  $$\#\,\mathrm{dom}(h) \leq 1$$

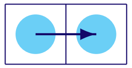
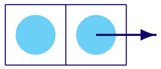
Configuration g with push (m, v) corresponds to dynamic configuration $(g, \{\, m \mapsto v \,\})$.

Mathematical Model for Shot Puzzles: Step

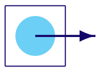
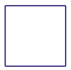
A dynamic configuration (g, h) where $h = \{ m \mapsto v \}$ evolves in one of three ways to a next dynamic configuration $\mathit{step}(g, h) = (g', h')$:

1.  If $g(m) + v \in G$ and $g(m) + v \notin \mathrm{ran}(g)$ then 

   $$(g', h') = (g \oplus \{ m \mapsto g(m) + v \}, h)$$

2.  If $g(m) + v \in G$ and $g(m) + v \in \mathrm{ran}(g)$, let $m' \in M$ such that $g(m') = g(m) + v$; then 

   $$(g', h') = (g, \{ m' \mapsto v \})$$

3.  If $g(m) + v \notin G$ then 

   $$(g', h') = (\{ m \} \lhd g, \varnothing)$$

Mathematical Model for Shot Puzzles: Move

If grid G is finite, then repeated (nested) application of *step* to dynamic configuration (g, h) eventually yields a next configuration of the form $(g', \varnothing)$.

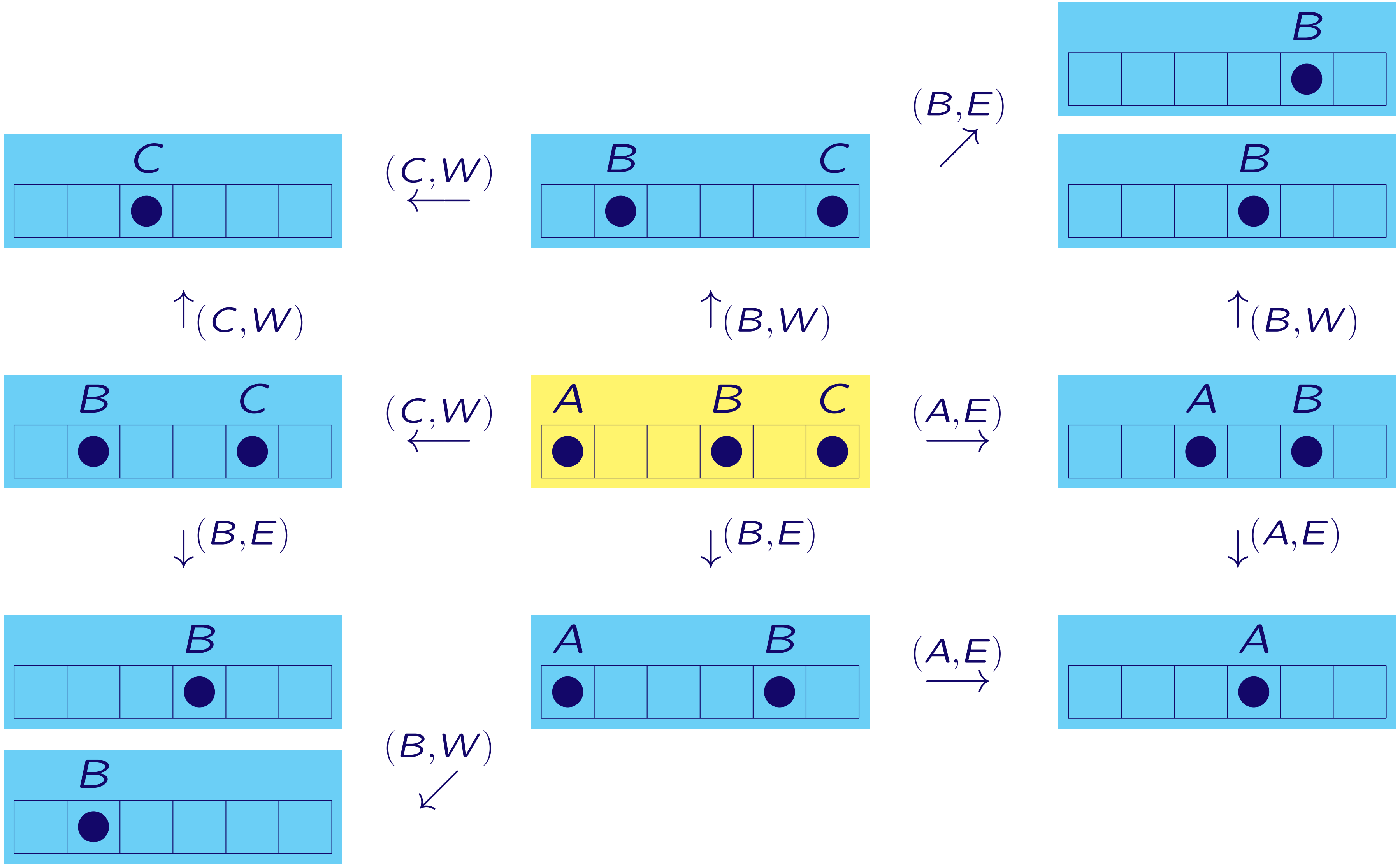
Repeated application of *step* is as a function *steps* from dynamic configurations to configurations, inductively defined by

$$\begin{aligned} steps(g, \varnothing) &= g \\ steps(g, \{ m \mapsto v \}) &= steps(step(g, \{ m \mapsto v \})) \end{aligned}$$

Note that $steps(g, h)$ indeed has one fewer marble than g.

Define $move(g, (m, v)) = steps(g, \{ m \mapsto v \})$ as the result of the move.

Mathematical Model for Shot Puzzles: Moves Example



Mathematical Model for Shot Puzzles: Objective

Reduce the set of marbles on the grid to a singleton by a sequence of moves.

Alternative Mathematical Model for Shot Puzzles

There are numerous, equivalent ways to model Shot puzzles formally.

Model development is a process with trade offs and design decisions.

Alternative: Simplify statics by omitting the grid.

A Shot puzzle is a pair (M, g), where

- M is a finite set (the marbles), and
- $g : M \rightarrowtail \mathbb{Z}^2$ is an injection from M to $\mathbb{Z} \times \mathbb{Z}$ (marble locations).

Complicates dynamics a bit (requiring quantifiers).

Alternative Mathematical Model for Shot Puzzles: Dynamics

Dynamic configuration (g, h) where $h = \{ m \mapsto v \}$ evolves in one of *two* ways to a next dynamic configuration $step(g, h) = (g', h')$:

1. *m* will hit *m′* If $\exists\, j : N^+ \bullet g(m) + jv \in \mathrm{ran}(g)$, then let

   $$k = min\{ j : \mathbb{N}^+ \mid g(m) + jv \in \mathrm{ran}(g) \bullet j \}$$

   and let $m' \in M$ such that $g(m') = g(m) + kv$; then

   $$(g', h') = (g \oplus \{ m \mapsto g(m) + (k-1)v \}, \{ m \} \lhd h \oplus \{ m' \mapsto v \})$$

2. *m* disappears If $\forall\, j : N^+ \bullet g(m) + jv \notin \mathrm{ran}(g)$, then

   $$(g', h') = (\{ m \} \lhd g, \varnothing)$$

Mathematical Model for Shot Puzzles: Validation

Formulate basic properties and prove them:

- A move reduces the number of marbles by one.
- Marble identity is irrelevant for solutions (isomorphism).
- The rules are symmetric under rotation over 90° and reflection.
- The size of a rectangular grid does not matter (for solvability).

Compare to unit test cases expressing expectations about a program.

Mathematical Model for Shot Puzzles: Reflection

- Implicitly defined state space
- More work than expected: Shot puzzles are not so trivial

  Two levels of granularity: $step$ and $move$
- More useful than expected: details matter and are discovered
- Easy to make mistakes
- Tool support desirable
- Z notation is very compact and usable
  (and I did not (yet) use Z schemas)

Enter Software

- How to describe concrete Shot puzzles? Need notation/language.
- How to process Shot puzzles?
- Ad hoc approach: define Shot-specific data types and I/O routines
- MDSE approach:
  - Define DSL and generate I/O routines
  - Define transformations to other domains, and use their tools

Domain Specific Languages

- Benefits:
  - Tooling available to support DSL development
  - DSL is independent of programming language
  - Syntax-aware editor for your DSL comes free
- Challenges:
  - Deal with semantics
  - Deal with (meta)model (co)evolution

DSL for Shot

- Abstract Syntax, the meta-model

  Compare to Abstract Data Type

- Eclipse Modeling Framework (EMF), using Ecore

  OCL for contraints

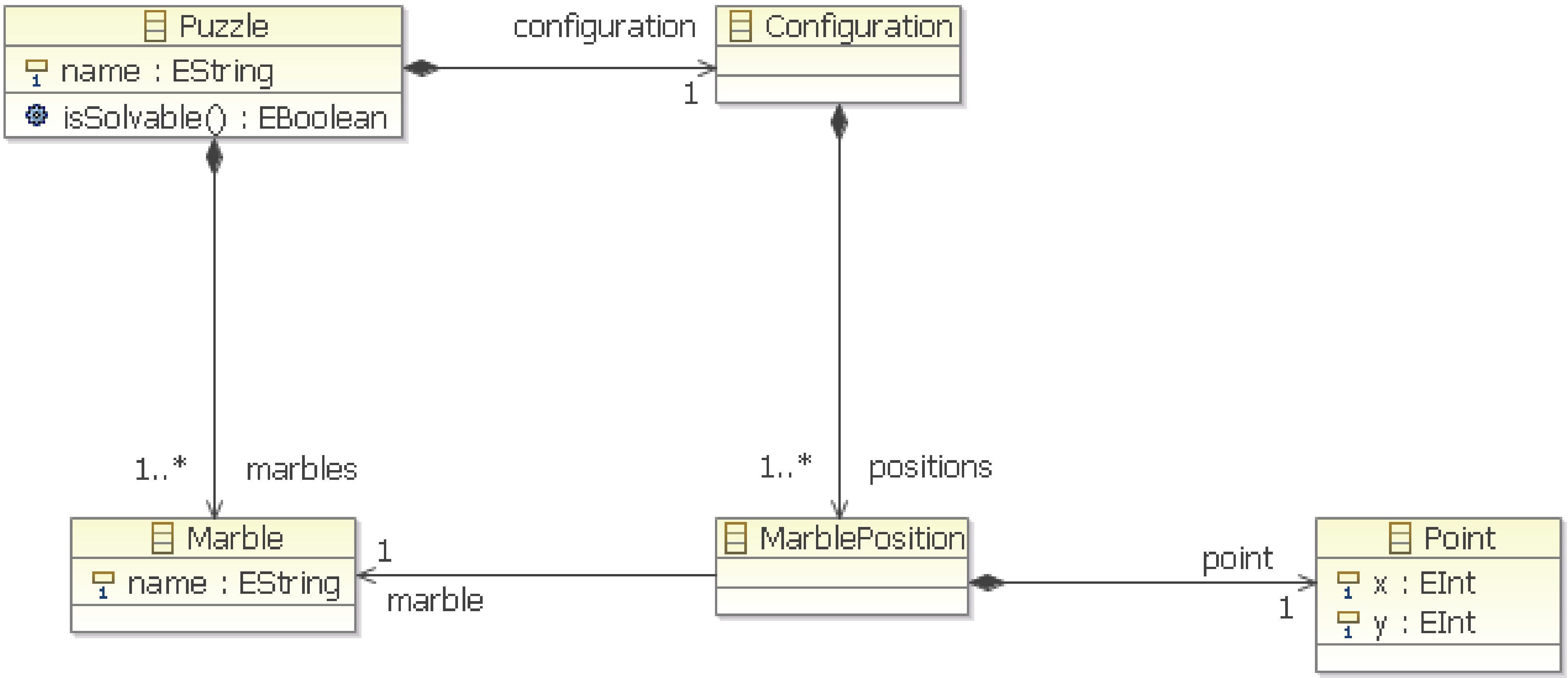
- Concrete Syntax (using EMFText)

- Semantics via meta-model extension of abstract syntax

  and model-to-model transformation

- Connect to other tools: via M2M, M2T, and T2M transformations

Abstract Syntax: Meta-Model for Statics



Meta-Model Derived Attributes

In EClass Configuration:

```
property marbles : Marble[*] { derived volatile }
{
    derivation: positions->collect(marble);
}
property points : Point[*] { derived volatile }
{
    derivation: positions->collect(point);
}
```

Meta-Model Constraints

In EClass Puzzle:

```
invariant ExactlyAllMarblesHaveInitialPosition:
    marbles->asBag() = configuration.marbles->asBag();

invariant MarbleNamesUnique:
    marbles->forAll( m1 : Marble, m2 : Marble
                   | m1.name = m2.name implies m1 = m2)
```

In EClass Configuration:

```
invariant MarblePositionsUnique:
    points->forAll( p1 : Point, p2 : Point
                  | p1.equals(p2) implies p1 = p2)
```

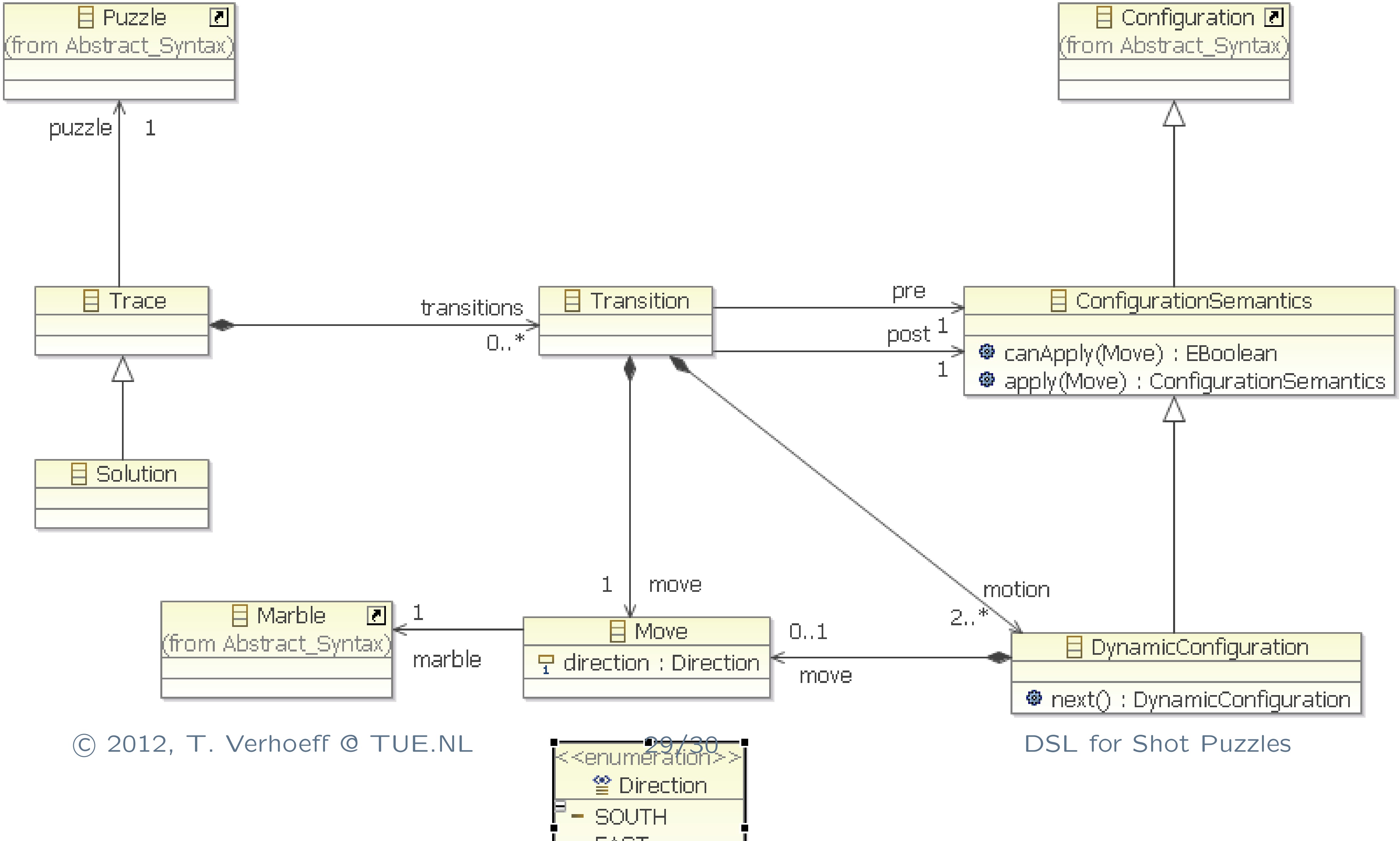
Conrete Syntax: Grammar (without pretty-printing info)

```
RULES {
    Puzzle ::=
        "Shot_puzzle" name[IDENTIFIER] "{"
            "marbles" marbles ("," marbles)* ";"
            configuration
        "}";

    Marble ::= name[IDENTIFIER];

    Configuration ::= "configuration" "{" (positions)+ "}";

    MarblePosition ::= marble[] "@" point ";";
    Point ::= "(" x[INTEGER] "," y[INTEGER] ")";
}
```

Concrete Syntax: Example of Textual Shot Puzzle Description

```
Shot_puzzle Easy {
    marbles A, B, C;
    configuration {
        A @ (0, 2);
        B @ (3, 2);
        C @ (5, 2);
    }
}
```

Meta-Model for Semantic Concepts (old version)



Conclusion

- Modeling concepts are usable to express semantics

  But: still useful to develop a mathematical model first

- EMF tools are quite usable, but still need further maturing

  Especially: Refactoring (cf. Refactory)