Symbolic Trajectories

Pattern Matching - Requirements and First Results

Fabio Valdés

20. März 2012

Contents

The operator `matches`

- Checks whether a pattern matches a moving label

The operator `matches`

- Checks whether a pattern matches a moving label
- Syntax:

The operator `matches`

- Checks whether a pattern matches a moving label
- Syntax:
  - $\texttt{mlabel} \times \texttt{pattern} \rightarrow \texttt{bool}$

The operator `matches`

- Checks whether a pattern matches a moving label
- Syntax:
  - `mlabel` × `pattern` → `bool`
  - `mlabel` × `text` → `bool`

Moving Label

Originally a stream of tuples of the form

```
( (instant start, instant end,
bool left_closed, bool right_closed)
ulabel unit_label)
```

Moving Label

Originally a stream of tuples of the form

```
( (instant start, instant end,
bool left_closed, bool right_closed)
ulabel unit_label)
```

Simplification

```
(periods period, ulabel unit_label)
```

Pattern

a simple example

```
(thursday _) (_ at_work) * (_ at_home)
(2011-04-02#19:09:00 _)
```

Pattern

a simple example

```
(thursday _) (_ at_work) * (_ at_home)
(2011-04-02#19:09:00 _)
```

variables may be associated to unit/sequence patterns

```
X (_ at_home) Y * Z (monday _)
```

Pattern

a simple example

```
(thursday _) (_ at_work) * (_ at_home)
(2011-04-02#19:09:00 _)
```

variables may be associated to unit/sequence patterns

```
X (_ at_home) Y * Z (monday _)
```

conditions can be added for variables

```
X (_ at_home) Y * Z (monday _) //
X.start > 2011-10-23, Y.card > 25,
Z.label = at_university
```

Unit Patterns

- `()`, an abbreviation for `(_ _)`, matches any unit

Unit Patterns

- `()`, an abbreviation for `(_ _)`, matches any unit
- the time interval can be entered in different ways

Unit Patterns

- `()`, an abbreviation for `(_ _)`, matches any unit
- the time interval can be entered in different ways
  - year, month, day, hour, minute, second

Unit Patterns

- `()`, an abbreviation for `(_ _)`, matches any unit
- the time interval can be entered in different ways
  - year, month, day, hour, minute, second
  - open or halfopen range of date or time

Unit Patterns

- `()`, an abbreviation for `(_ _)`, matches any unit
- the time interval can be entered in different ways
  - year, month, day, hour, minute, second
  - open or halfopen range of date or time
  - semantic time range

Sequence Patterns

- $*$ matches any sequence

Sequence Patterns

- $*$ matches any sequence
- $+$ matches any sequence with at least one unit

Sequence Patterns

- `*` matches any sequence
- `+` matches any sequence with at least one unit
- `((time label))` matches a continuous sequence of units that fulfill the condition(s)

Features to be Added

- Regular Expressions. If p_1 and p_2 are patterns, then

  are also patterns

Features to be Added

- Regular Expressions. If p_1 and p_2 are patterns, then
  - $[\, p_1 \mid p_2 \,]$

are also patterns

Features to be Added

- Regular Expressions. If p_1 and p_2 are patterns, then
  - $[\, p_1 \mid p_2 \,]$
  - $[\, p_1 \,] +$

  are also patterns

Features to be Added

- Regular Expressions. If p_1 and p_2 are patterns, then
  - $[\, p_1 \mid p_2 \,]$
  - $[\, p_1 \,]+$
  - $[\, p_1 \,]*$

  are also patterns

Features to be Added

- Regular Expressions. If p_1 and p_2 are patterns, then
  - $[\, p_1 \mid p_2 \,]$
  - $[\, p_1 \,]+$
  - $[\, p_1 \,]*$

  are also patterns
- Label Hierarchies

Features to be Added

- Regular Expressions. If p_1 and p_2 are patterns, then
  - $[\, p_1 \mid p_2 \,]$
  - $[\, p_1 \,]+$
  - $[\, p_1 \,]*$

  are also patterns
- Label Hierarchies
  - e.g., `leisure` $\rightarrow$ `building` $\rightarrow$ `museum` $\rightarrow$ `...`

The Operator `apply`

- Syntax:

The Operator `apply`

- Syntax:
  - `mlabel` × `rule` → `stream`(`mlabel`)

The Operator `apply`

- Syntax:
  - `mlabel` $\times$ `rule` $\rightarrow$ `stream(mlabel)`
  - `mlabel` $\times$ `text` $\rightarrow$ `stream(mlabel)`

The Operator `apply`

- Syntax:
  - `mlabel` × `rule` → `stream`(`mlabel`)
  - `mlabel` × `text` → `stream`(`mlabel`)
- For each way the pattern matches, one rewritten version of it is returned

The Operator `apply`

- Syntax:
  - `mlabel` × `rule` → `stream`(`mlabel`)
  - `mlabel` × `text` → `stream`(`mlabel`)
- For each way the pattern matches, one rewritten version of it is returned
- If there is no match, the stream is empty