The algorithm of context recognition in TIL

Marie Duží, Michal Fait

Goals

- Implementation of the algorithm recognizing three kinds of context in TIL using Prolog language
  - System processes computational variant of TIL, the TIL-Script language
  - The output is a *derivation tree* of TIL-Script constructions with their *occurrence* written in XML
  - Beside context recognition the system also performs *syntactic* and *type control* of TIL-Script constructions

Contexts in TIL

Context recognition enables us to determine how to ***correctly apply basic inference rules***

Václav Klaus is the president of CR
The president of CR is an economist
_____ *valid*
Václav Klaus is an economist

Václav Klaus is the president of CR
Jan Sokol wants to become the president of CR
_____ ***in****valid*
Jan Sokol wants to become Václav Klaus.

Levels of abstractions

- Operations on ***procedures***
  - Construction is mentioned by **Trivialisation** (occurs in the ***displayed mode***)
  - **Construction** itself is an argument of another function

  → ***Hyperintensional context***
- Operations on the ***product*** of a procedure
  - Construction occurs in the ***execution mode***, and we distinguish:
    - ***Intensionally*** – the whole constructed ***function*** is an object of predication
    - ***Extensionally*** – the ***value*** (if any) of the constructed function is an object of predication

Context recognition

Our goal is to recognize in which way a subconstruction *C* of a construction *D occurs*

- Hyperintensional context is *dominant* over intensional and extensional ones
  - All the subconstructions occurring within a Trivialized construction occur hyperintensionally, unless the effect of Trivialization is cancelled by Double execution: $^{20}C = C$
- Intensional λ-generic context is *dominant* over an extensional occurrence

Intensional/extensional occurrences of a constituent in *C*

Construction C	*Occurrence in C*	
	intensional	*Extensional*
$[X_0 X]$	$[X_0 X]$, X	X_0
$\lambda x_1\ [X_0 X]$	$\lambda x_1 [X_0 X]$, $[X_0 X]$, X_0, X, all constituents	–
$[\lambda x_1\ [X_0 X]\ X_1]$	$[\lambda x_1 [X_0 X]\ X_1]$, $[X_0 X]$, X, X_1	$\lambda x_1\ [X_0 X]$, X_0
$\lambda x_2\ [\lambda x_1\ [X_0 X]]$	all constituents	–
$[\lambda x_2\ [\lambda x_1\ [X_0 X]]\ X_2]$	$[\lambda x_2\ [\lambda x_1\ [X_0 X]]\ X_2]$, $[\lambda x_1 [X_0 X]]$, $[X_0 X]$, X, X_2	$\lambda x_2\ [\lambda x_1\ [X_0 X]]$
$[\lambda x_2\ [\lambda x_1\ [X_0 X]\ X_1]]$	all constituents	–
$[\lambda x_2\ [\lambda x_1\ [X_0 X]\ X_1]\ X_2]$	$[\lambda x_2\ [\lambda x_1\ [X_0 X]\ X_1]\ X_2]$, $[X_0 X]$, X, X_1, X_2	$\lambda x_2 [\lambda x_1 [X_0 X]\ X_1]$, $\lambda x_1 [X_0 X]$, X_0
$\lambda x_1\ x_2\ [X_0 X]$	$\lambda x_1\ x_2\ [X_0 X]$, X_0, X, all constituents	–
$[\lambda x_1\ x_2\ [X_0 X]\ X_1\ X_2]$	$[\lambda x_1\ x_2\ [X_0 X]\ X_1\ X_2]$, $[X_0 X]$, X, X_1, X_2	$\lambda x_1\ x_2\ [X_0 X]$, X_0

Implementation

- TIL-Script constructions serve as an input to the system, the output is an XML file where the occurrences (hyperintensional, intensional and extensional) of particular subconstructions are marked out
- Processing input file is implemented in C#, context recognition, type control, and output into XML file are implemented in Prolog language

Processing in Prolog

- Output constructions from syntactic analysis are reworked in Prolog
- For every construction a record is created (relation *construction/10*), which contains all necessary information
- Records with mutual bounds create a tree structure

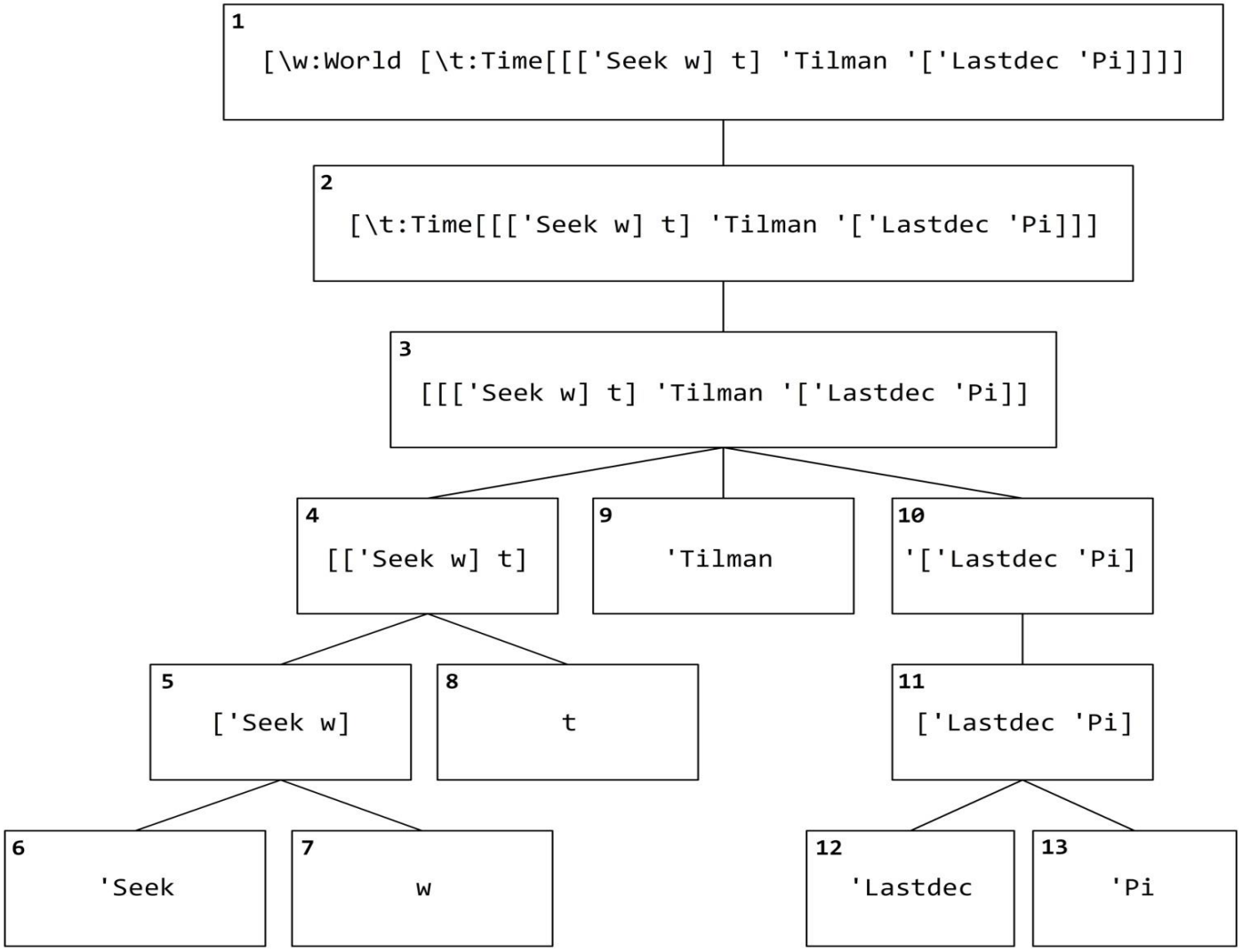
Depth-first *search*

- The algorithms that process constructions in Prolog (type control or Hyperintensional context recognition), apply the *depth-first search strategy*

Name: depthFirstSearch
Input: construction C

Process construction C
P = child constructions of construction C
For every construction D in P do
    depthFirstSearch(D)
Process construction C

Example of tree for construction [\w: World [\t:Time['Seek@wt 'Tilman '['Lastdec 'Pi]]]],
Nodes are numbered in order to visit them during Depth-first *search* .



Hyperintensional context recognition

- For the hyperintensional context recognition the depth-first search strategy is applied
- When we reach Trivialisation of a construction, in the process of traversing its subconstructions we mark out hyperintensional context
- When we reach Double Execution ^{2}C where C is Trivialisation ^{0}D, the effect of this Trivialisation is cancelled → *D* occurs as an (intensional/extensional) *constituent* rather than hyperintensionally

Hyperintensional context recognition; *algorithm*

Name: determineHyperintensional

Input: construction C, constant S indicating, whether current construction is used or mentioned

If S=”mentioned”

Save hyperintensional context of construction C

If S=”used” and C is trivialisation of construction ^{0}D

call determineHyperintensional(D,”mentioned”)

else

If S=”used” and C is double execution ^{2}D and D is trivialization of construction ^{0}E

call determineHyperintensional(E,”used”)

Else

P = child constructions of construction C

For every construction X in P do

determineHyperintensional (X,S)

Extensional context recognition

- Consists of two steps
  - finding out which constructions occur with extensional supposition and then
  - check if there is no dominating λ-generic intensional context

Extensional context recognition

- Using the consequences of the definition of extensional/intensional supposition, we derive this algorithm

Name: extSupositionConstructions
Output: set of constructions with extensional suposition

For every composition C in form $[X\ Y_1 \ldots Y_m]$ do
    If C does not occur in hyperintensional context
        Add construction X to the result
For every execution E in form ^{1}X or ^{2}X, where X is object of order one do
    If E does not occur in hyperintensional context
        Add construction E to the result
For every execution E in form ^{2}X, where X v-constructs object of order one do
    If E does not occur in hyperintensional context
        Add construction E to the result

Extensional context recognition

- Now we have to check genericity;
  - the algorithm follows the same recursive way as the formal definition specifies

- If a construction with the extensional supposition occurs in a non–generic context, we can mark out extensional occurrence

Intensional context recognition

- The last step is simple.
- For every construction do:
  - if the occurrence is not hyperintensional or extensional, then it is an intensional occurrence
  - Hence *intensional context*

construction [\w:World [\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]]

```
<construction occurrence ="Intensional" construction ="[\w:World [\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]]">
 <construction occurrence ="Intensional" construction ="[\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]">
   <construction occurrence ="Intensional" construction ="[[['Seek w] t] 'Tilman '['Lastdec 'Pi]]">
     <construction occurrence ="Extensional" construction ="[['Seek w] t]">
       <construction occurrence ="Extensional" construction ="['Seek w]">
         <construction occurrence ="Extensional" construction ="'Seek"></construction>
         <construction occurrence ="Intensional" construction ="w"></construction>
       </construction>
       <construction occurrence ="Intensional" construction ="t"></construction>
     </construction>
     <construction occurrence ="Intensional" construction ="'Tilman"></construction>
     <construction occurrence ="Intensional" construction ="'['Lastdec 'Pi]">
       <construction occurrence ="Hyperintensional" construction ="['Lastdec 'Pi]">
         <construction occurrence ="Hyperintensional" construction ="'Lastdec"></construction>
         <construction occurrence ="Hyperintensional" construction ="'Pi"></construction>
       </construction>
      </construction>
    </construction>
  </construction>
</construction>
```

Output into console

Construction [\w:World [\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]]

Intensional: [\w:World [\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]]
Intensional: [\t:Time [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]]
Intensional: [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]
Intensional: [['Seek w] t]
Intensional: ['Seek w]
Intensional: 'Seek
Intensional: w
Intensional: t
Intensional: 'Tilman
Intensional: '['Lastdec 'Pi]
Hyperintensional: ['Lastdec 'Pi]
Hyperintensional: 'Lastdec
Hyperintensional: 'Pi

Output into console

Construction [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]

Intensional: [[['Seek w] t] 'Tilman '['Lastdec 'Pi]]
Extensional: [['Seek w] t] (de re)
Extensional: ['Seek w] (*de re*)
Extensional: 'Seek (*de re*)
Intensional: w
Intensional: t
Intensional: 'Tilman
Intensional: '['Lastdec 'Pi]
Hyperintensional: ['Lastdec 'Pi]
Hyperintensional: 'Lastdec
Hyperintensional: 'Pi

Thank you for your attention