Graphical Representation for Global Protocols

Charlotte Pichot,
MSc in Advanced Computing
Supervisor: Dr. Nobuko Yoshida
Department of Computing, Imperial College London

Imperial College London

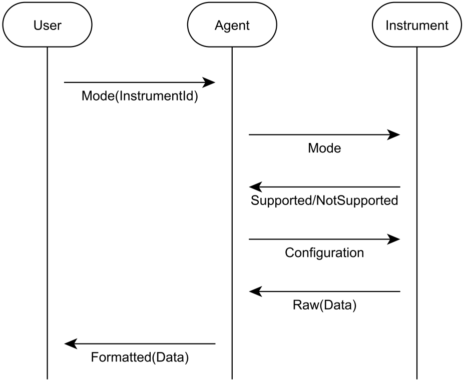
Contents

1. Motivations
    - Example
    - Generalised Multiparty Session Types
    - Overview of the project
2. Graph Representation
    - The design
    - Syntax
    - Results
3. About the Implementation
    - Structure of the development
    - Demonstration

Contents

1. Motivations
   - Example
   - Generalised Multiparty Session Types
   - Overview of the project
2. Graph Representation
   - The design
   - Syntax
   - Results
3. About the Implementation
   - Structure of the development
   - Demonstration

Ocean Observatories Initiative Use Case : DataAcquisition



Global Protocol in SCRIBBLE

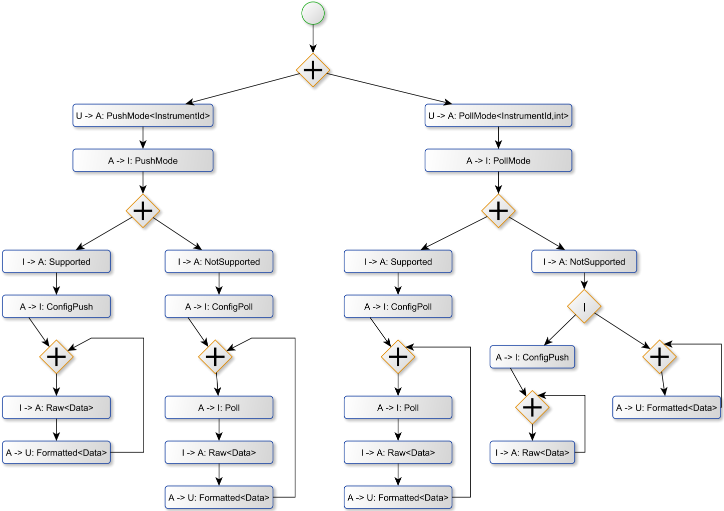
```
// U is User, A is ION Agent (Integrated
// Observatories Network), I is Instrument

global protocol DataAcquisition (role U, role A, role I) {

interruptible { choice at U {
    PushMode(InstrumentId) from U to A;
    PushMode from A to I;
    choice at I {
        Supported from I to A;
        ConfigPush from A to I;
        rec PUSH {
            Raw(Data) from I to A;
            Formatted(Data) from A to U;
            continue  PUSH;}
    } or {NotSupported from I to A;
        ConfigPoll from A to I;
        rec POLL {
            Poll from A to I;
            Raw(Data) from I to A;
            Formatted(Data) from A to U;
            continue  POLL;} }
} or {   PollMode(InstrumentId, int) from U to A;
    PollMode from A to I;
    choice at I {
        Supported from I to A;
        ConfigPoll from A to I;
        rec POLL {
            Poll from A to I;
            Raw(Data) from I to A;
            Formatted(Data) from A to U;
            continue POLL;}
    } or {NotSupported from I to A;
        parallel {
            ConfigPush from A to I;
            rec PUSH {
                Raw(Data) from I to A;
                continue PUSH;}
        } and {     rec POLL {
                Formatted(Data) from A to U;
                continue POLL; }} }}

} by U with Stop
}
```

Graphical Representation

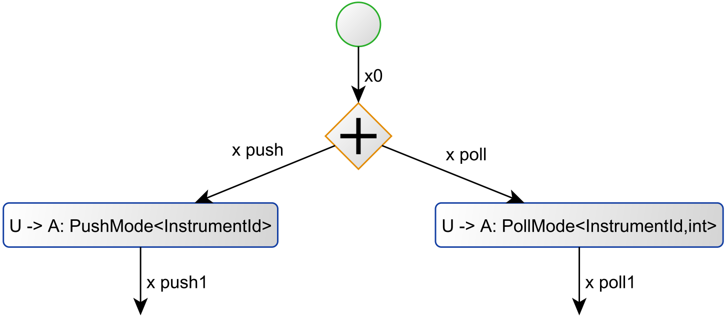


Generalised Multiparty Session Types: Global Types

$$
\begin{array}{lcll}
\mathrm{G} & ::= & \mathrm{def}\ G\ \mathrm{in}\ \mathrm{x} & \text{Global type} \\
G & ::= & \mathrm{x} = \mathrm{p} \to q : l\langle U\rangle ; x' & \text{Labelled messages} \\
 & | & \mathrm{x} = \mathrm{x'} \mid \mathrm{x''} & \text{Fork} \\
 & | & \mathrm{x} = \mathrm{x'} + \mathrm{x''} & \text{Choice} \\
 & | & \mathrm{x} \mid \mathrm{x'} = \mathrm{x''} & \text{Join} \\
 & | & \mathrm{x} + \mathrm{x'} = \mathrm{x''} & \text{Merge} \\
 & | & \mathrm{x} = \mathrm{end} & \text{End} \\
U & ::= & \langle G\rangle \mid \mathrm{bool} \mid nat \mid \ldots & \text{Sorts}
\end{array}
$$

$$
\begin{array}{llrcl}
\mathrm{G} = & \mathrm{def} & x_0 & = & x_{push} + x_{poll} \\
 & & x_{push} & = & \mathrm{U} \to \mathrm{A}\text{: PushMode}\langle\ \text{InstrumentId}\ \rangle; x_{push1} \\
 & & x_{push1} & = & \mathrm{A} \to \mathrm{I}\text{: PushMode }; x_{push2} \\
 & & x_{push2} & = & x_{ps} + x_{pns} \\
 & & x_{ps} & = & \mathrm{I} \to \mathrm{A}\text{: Supported}; x_{ps1} \\
 & & x_{ps1} & = & \mathrm{A} \to \mathrm{I}\text{: ConfigPush}; x_{ps2} \\
 & & x_{ps2} + x_{ps3} & = & x_{ps4} \\
 & & x_{ps4} & = & \mathrm{I} \to \mathrm{A}\text{: Raw}\langle\ \text{Data}\ \rangle; x_{ps5} \\
 & & x_{ps5} & = & \mathrm{A} \to \mathrm{U}\text{: Formatted}\langle\ \text{Data}\ \rangle; x_{ps3} \\
 & & x_{pns} & = & \mathrm{I} \to \mathrm{A}\text{: NotSupported}; x_{pns1} \\
 & & x_{pns1} & = & \mathrm{A} \to \mathrm{I}\text{: ConfigPoll}; x_{pns2}\ \ldots \\
 & \mathrm{in}\ x_0 & & &
\end{array}
$$

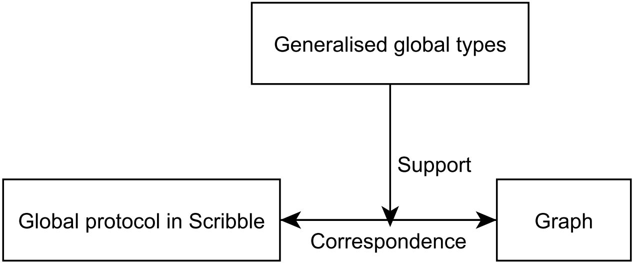
Generalised Multiparty Session Types: Graph syntax


x0
x push
x poll
U -> A: PushMode<InstrumentId>
U -> A: PollMode<InstrumentId,int>
x push1
x poll1

```
choice at U {
  PushMode(InstrumentId) from U to A;
} or {
  PollMode(InstrumentId,int) from U to A;
}
```

$$
\begin{aligned}
x_0 &= x_{push} + x_{poll} \\
x_{push} &= \mathrm{U} \rightarrow \mathrm{A}\text{: PushMode}\langle\ \text{InstrumentId}\ \rangle; x_{push1} \\
x_{poll} &= \mathrm{U} \rightarrow \mathrm{A}\text{: PollMode}\langle\ \text{InstrumentId,int}\ \rangle; x_{poll1}
\end{aligned}
$$

Overview of the project



Generalised global types

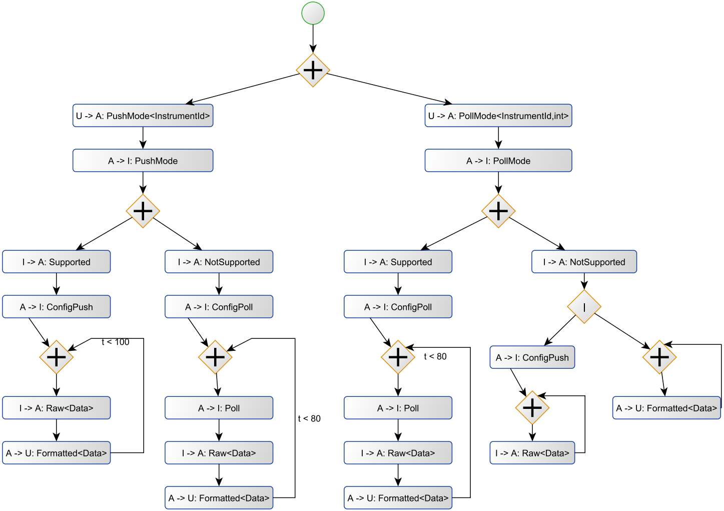
Support

Global protocol in Scribble

Correspondence

Graph

Graphical Representation with time constraints



+

U -> A: PushMode<InstrumentId>

U -> A: PollMode<InstrumentId,int>

A -> I: PushMode

A -> I: PollMode

+

+

I -> A: Supported

I -> A: NotSupported

I -> A: Supported

I -> A: NotSupported

A -> I: ConfigPush

A -> I: ConfigPoll

A -> I: ConfigPoll

|

+

t < 100

+

+

t < 80

A -> I: ConfigPush

+

+

t < 80

I -> A: Raw<Data>

A -> I: Poll

A -> I: Poll

A -> U: Formatted<Data>

A -> U: Formatted<Data>

I -> A: Raw<Data>

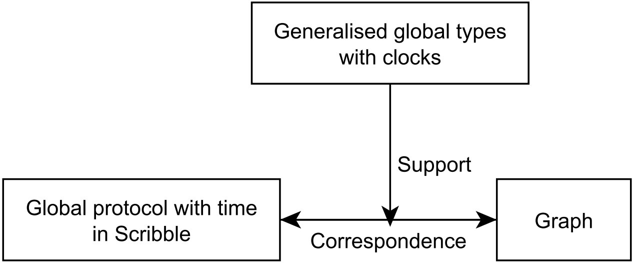
I -> A: Raw<Data>

I -> A: Raw<Data>

A -> U: Formatted<Data>

A -> U: Formatted<Data>

Overview of the project



Generalised global types with clocks

Support

Global protocol with time in Scribble

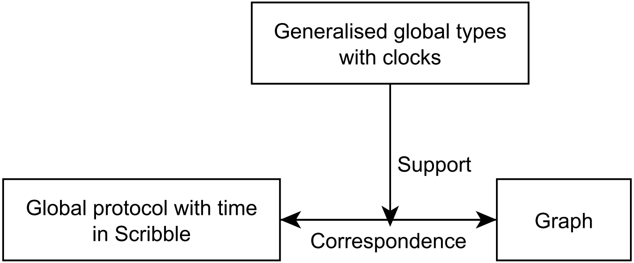
Correspondence

Graph

Contributions:

- Design of the graph
- Extension with clocks
- Implementation of the correspondence

Overview of the project



Generalised global types with clocks

Support

Global protocol with time in Scribble

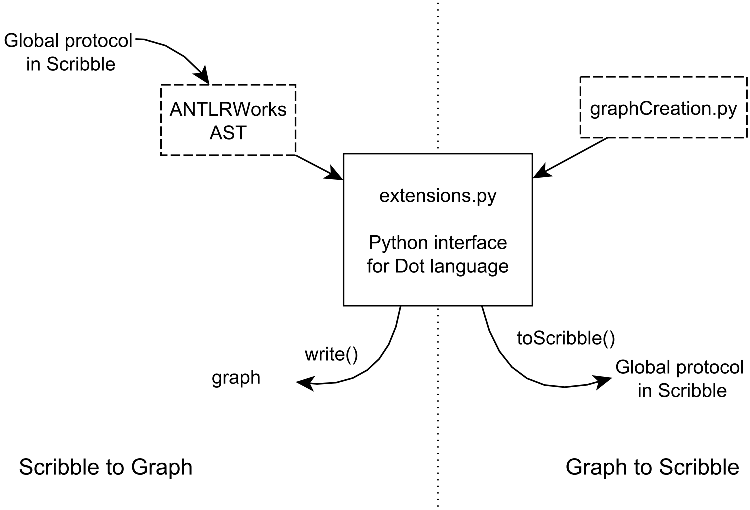
Correspondence

Graph

Contributions:

- Design of the graph
- Extension with clocks
- Implementation of the correspondence

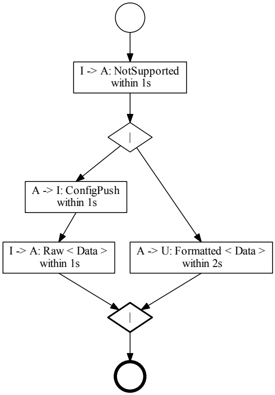
Overview of the implementation



Contents

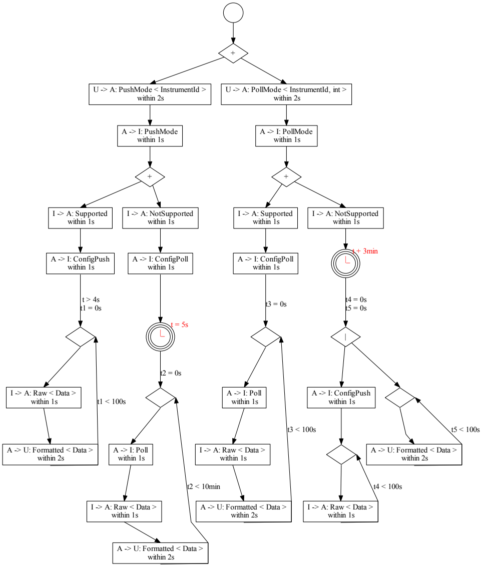
1. Motivations
   - Example
   - Generalised Multiparty Session Types
   - Overview of the project
2. Graph Representation
   - The design
   - Syntax
   - Results
3. About the Implementation
   - Structure of the development
   - Demonstration

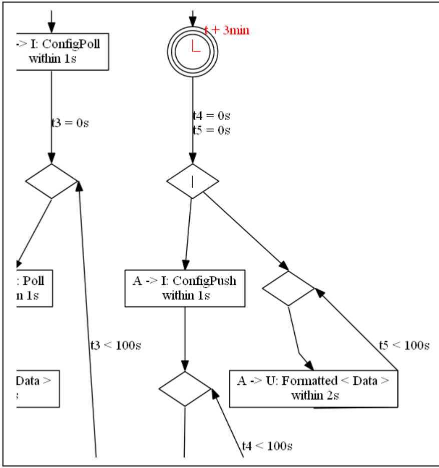
Graphical notations



```
global protocol FirstParallel (role U, role A, role I) {
 NotSupported from I to A within 1s;
 parallel {
    ConfigPush from A to I within 1s;
    Raw(Data) from A to I within 1s;
 } and {
    Formatted(Data) from A to U within 2s;
    }
}
```

Graph with clocks





Zoom

Timed global types

Let C be a set of clocks. C = $\{t, t_1, t_2, ..., t_x, ...\}$

Definition:

$\delta := t \leq v \mid t \geq v \mid \neg\delta \mid \delta_1 \wedge \delta_2 \mid \varepsilon$
where t is a clock in C and v is a constant in $\mathbb{Q}$.

Abbreviations:

t = v	means	t $\leq$ v and t $\geq$ v
t < v	means	$\neg$ t $\geq$ v
t > v	means	$\neg$ t $\leq$ v

$$
\begin{array}{llll}
\mathrm{G} & ::= & \text{def } \tilde{G} \text{ in x} & \text{Global type} \\
G & ::= & \text{x = p} \rightarrow p' : l\langle U\rangle, \lambda_O, \delta_O, \lambda_I, \delta_I; x' & \text{Labelled messages} \\
 & \mid & \text{x = x' | x''} & \text{Fork} \\
 & \mid & \text{x = x' + x''} & \text{Choice} \\
 & \mid & \text{x | x' = x''} & \text{Join} \\
 & \mid & \text{x + x' = x''} & \text{Merge} \\
 & \mid & \text{x = end} & \text{End} \\
U & ::= & \langle G\rangle \mid \text{bool} \mid nat \mid \ldots & \text{Sorts}
\end{array}
$$

Timed local types and projection

T	::=	def $\tilde{T}$ in x	Local type
T	::=	$x = !\langle p, l\langle U\rangle, \lambda, \delta\rangle.x'$	Message sending
	|	$x = ?\langle p, l\langle U\rangle, \lambda, \delta\rangle.x'$	Message receiving
	|	$x = x' \mid x''$	Fork
	|	$x = x' \oplus x''$	Internal choice
	|	$x = x' \,\&\, x''$	External choice
	|	$x \mid x' = x''$	Join
	|	$x + x' = x''$	Merge
	|	$x = x'$	Inaction
	|	$x = \mathrm{end}$	End

Projection algorithm:

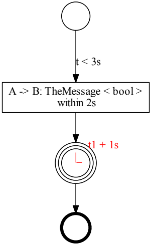
$$\mathrm{def}\ \tilde{G}\ \mathrm{in}\ x \upharpoonright p \quad = \quad \mathrm{def}\ \tilde{G} \upharpoonright_{\tilde{G}} p\ \mathrm{in}\ x$$

$$\begin{array}{rcll}
x = p \to p' : l\langle U\rangle, \lambda_O, \delta_O, \lambda_I, \delta_I; x' \upharpoonright_{\tilde{G}} p & = & x = !\langle p', l\langle U\rangle, \lambda_O, \delta_O\rangle.x' & \\
x = p \to p' : l\langle U\rangle, \lambda_O, \delta_O, \lambda_I, \delta_I; x' \upharpoonright_{\tilde{G}} p' & = & x = ?\langle p, l\langle U\rangle, \lambda_I, \delta_I\rangle.x' & \\
x = p \to p' : l\langle U\rangle, \lambda_O, \delta_O, \lambda_I, \delta_I; x' \upharpoonright_{\tilde{G}} p'' & = & x = x' & (p'' \notin \{p,p'\}) \\
x = x' \mid x'' \upharpoonright_{\tilde{G}} p & = & x = x' \mid x'' & \\
x = x' + x'' \upharpoonright_{\tilde{G}} p & = & x = x' \oplus x'' & (\text{if } p = \mathrm{ASend}(\tilde{G})(x)) \\
x = x' + x'' \upharpoonright_{\tilde{G}} p & = & x = x' \,\&\, x'' & (\text{otherwise}) \\
x \mid x' = x'' \upharpoonright_{\tilde{G}} p & = & x \mid x' = x'' & \\
x + x' = x'' \upharpoonright_{\tilde{G}} p & = & x + x' = x'' & \\
x = \mathrm{end} \upharpoonright_{\tilde{G}} p & = & x = \mathrm{end} &
\end{array}$$

Timed global types: Example

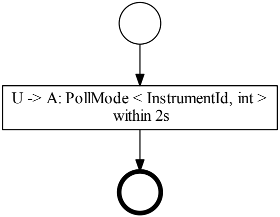
$$x = p \rightarrow p' : l\langle U\rangle, \lambda_O, \delta_O, \lambda_I, \delta_I \ ;x'$$

$$x = A \rightarrow B : \text{TheMessage <bool>} , \{t_x\}, t < 3, \{t_1\}, t_x < 2 \ ; x'$$



```
global protocol Example (role A, role B) {
  t before 3s
 TheMessage(bool) from A to B within 2s;
 wait for t1+ 1s
}
```

The *within* statement



```
global protocol TestMessage (role A, role U) {
  PollMode(InstrumentId,int) from U to A within 2s;
}
```

To support this protocol we define the global type as follows:

$G =$ def x = U $\rightarrow$ A : PollMode <InstrumentId, int> , $\{t_x\}, \varepsilon, \emptyset, t_x < 2$; x'
x' = end
in x

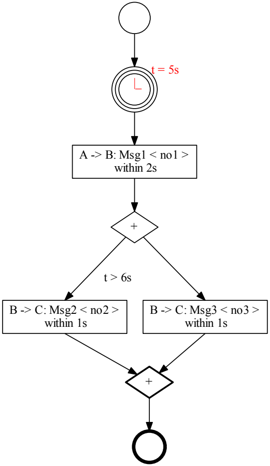
$T_U =$ def x = !$\langle$ A, PollMode <InstrumentId, int> ,$\{t_x\}, \varepsilon\rangle$. x'
x' = end
in x

$T_A =$ def x = ?$\langle$ U,PollMode <InstrumentId, int> , $\emptyset, t_x < 2\rangle$. x'
x' = end
in x

Temporal satisfiability for clocks conditions

Temporal satisfiability *If δ, the clock condition of a given transition, is satisfiable at some point, for each constraint δ', appearing in a later transition, it is eventually possible to satisfy δ'.*

```
global protocol TestTemporalSatisfiability (role A, role
B, role C) {
 wait for t is 5s
 Msg1(no1) from A to B within 2s;
    choice at B {
       t after 6s
       Msg2(no2) from B to C within 1s;
    } or {
       Msg3(no3) from B to C within 1s;
    } }
```



Timed processes

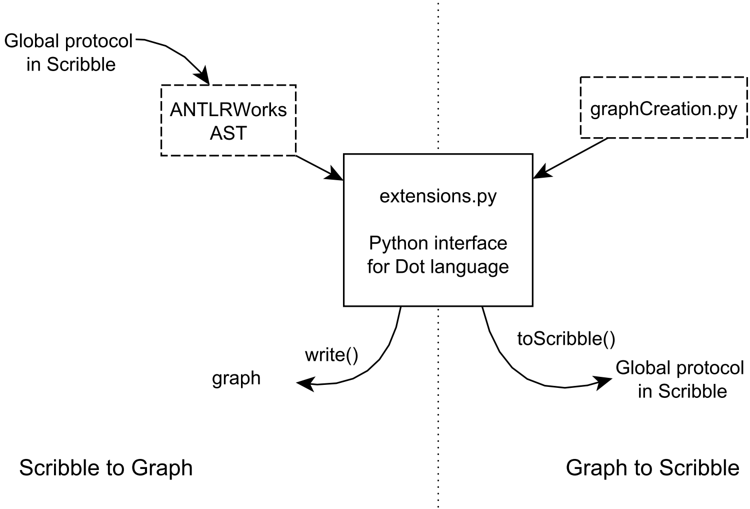
P	::=	def $\tilde{P}$ in X	definition
P	::=	$x(\tilde{x}) = x\langle G, C\rangle.x'(\tilde{e})$	init
	$\mid$	$x(\tilde{x}) = x[p](y) : x'(\tilde{e})$	accept
	$\mid$	$x(\tilde{x}) = x!\langle p; l < e >, \lambda_O, \delta_O\rangle : x'(\tilde{e})$	send
	$\mid$	$x(\tilde{x}) = x?\langle p; l(y), \lambda_I, \delta_I\rangle : x'(\tilde{e})$	receive
	$\mid$	$x(\tilde{x}) = x'(\tilde{y}) \mid x''(\tilde{z})$	parallel
	$\mid$	$x(\tilde{x}) \mid x'(\tilde{y}) = x''(\tilde{z})$	join
	$\mid$	$x(\tilde{x}) + x'(\tilde{x}) = x''(\tilde{x})$	merge
	$\mid$	$x(\tilde{x}) = x'(\tilde{x})$ & $x''(\tilde{x})$	external choice
	$\mid$	if e then $x'(\tilde{e}')$ else $x''(\tilde{e}'')$	conditional
	$\mid$	$x(\tilde{x}) = 0$	null
	$\mid$	$x(\tilde{x}) = (\nu a)x'(a\tilde{x})$	new name
X	::=	$x(\tilde{v}) \mid X \mid X$	thread, parallel
	$\mid$	$x(\nu a)X \mid 0$	restriction, null
e	::=	$v \mid x \mid e \wedge \delta \mid e \wedge e \mid \ldots$	expressions
v	::=	$a \mid s[p] \mid \text{true} \mid \text{false} \mid \ldots$	values
α, β	::=	$s[p,q]!\langle p; l < e >, \lambda, \delta\rangle$	labels
	$\mid$	$s[p,q]?\langle p; l(y), \lambda, \delta\rangle$	
	$\mid$	$a\langle G, C\rangle \mid a\langle p\rangle\ [s] \mid \langle\tau, \lambda, \delta\rangle$	

Table: Syntax for timed processes

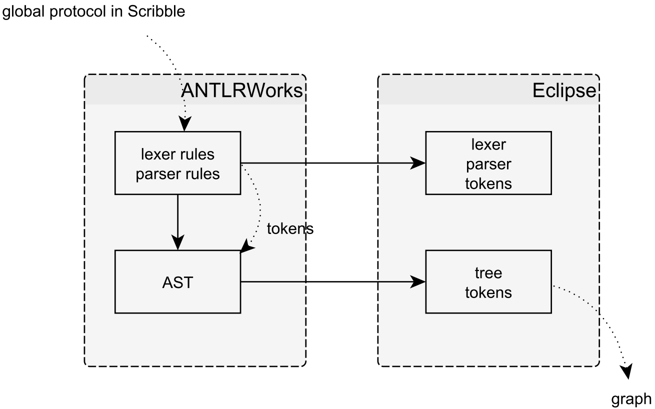
Contents

1. Motivations
   - Example
   - Generalised Multiparty Session Types
   - Overview of the project
2. Graph Representation
   - The design
   - Syntax
   - Results
3. About the Implementation
   - Structure of the development
   - Demonstration

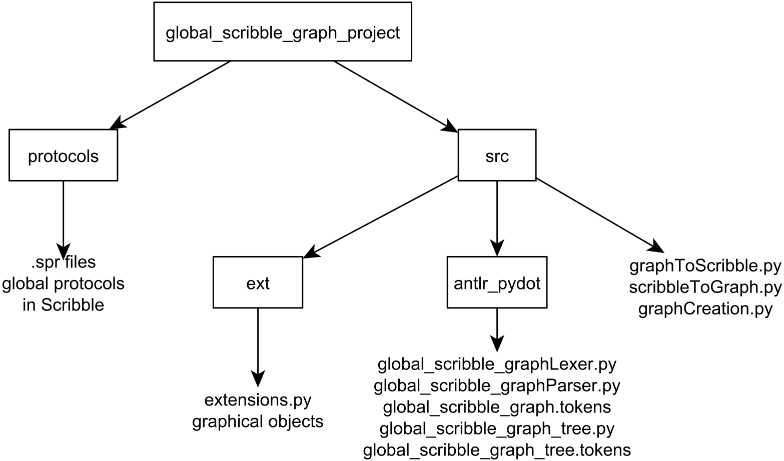
Overview


Global protocol in Scribble
ANTLRWorks AST
graphCreation.py
extensions.py
Python interface for Dot language
write()
graph
toScribble()
Global protocol in Scribble
Scribble to Graph
Graph to Scribble

Link between ANTLRWorks and Eclipse



Demonstration


global_scribble_graph_project
protocols
src
.spr files
global protocols
in Scribble
ext
antlr_pydot
graphToScribble.py
scribbleToGraph.py
graphCreation.py
extensions.py
graphical objects
global_scribble_graphLexer.py
global_scribble_graphParser.py
global_scribble_graph.tokens
global_scribble_graph_tree.py
global_scribble_graph_tree.tokens

Future work

- Well-formedness verification
- Further extensions of Scribble: merge, join, etc.
- Merge with Guillaume's project about the plugin Eclipse
- Proofs of the properties for clocks condition and timed processes

Future work

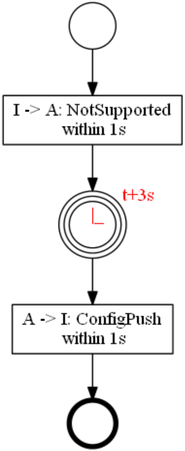
- Well-formedness verification
- Further extensions of Scribble: merge, join, etc.
- Merge with Guillaume's project about the plugin Eclipse
- Proofs of the properties for clocks condition and timed processes

Questions

Extended Scribble language

```
       global-protocol-body  ::=  global-interaction-block
   global-interaction-block  ::=  { global-interaction-sequence }
global-interaction-sequence  ::=  ( global-interaction )*
         global-interaction  ::=  [ time-constraints ] message
                              |   [ time-constraints ] choice
                              |   [ time-constraints ] parallel
                              |   [ time-constraints ] recursion
                              |   [ time-constraints ] continue
                              |   [ time-constraints ] delay
                    message  ::=  ( message-signature | identifier) from role-name
                                  to role-name within time;
                      delay  ::=  wait  for time-identifier symbol time ;
                              |   wait  for time-identifier is time ;
           time-constraints  ::=  constraint (and constraint )*
                 constraint  ::=  time-identifier after time
                              |   time-identifier before time
                              |   time-identifier is time
            time-identifier  ::=  identifier
                       time  ::=  ( digit )* identifier
                     symbol  ::=  ( ‘+’ | ’*’ )
```

Delay

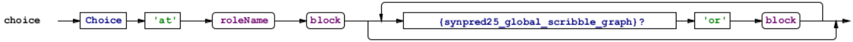


```
global protocol FirstDelay (role A, role I) {
  NotSupported from I to A within 1s;
  wait for t+ 3s
  ConfigPush from A to I within 1s;
}
```

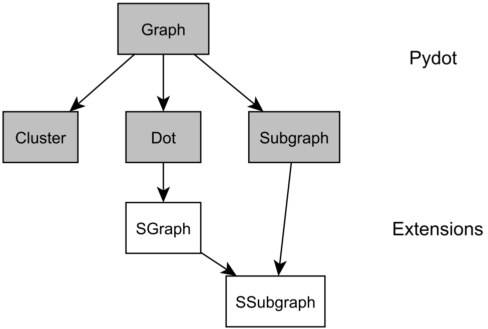
How to write grammar with ANTLR?

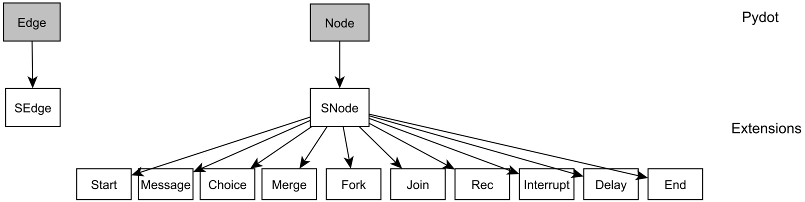
```
choice at U {
  PushMode(InstrumentId) from U to A;
} or {
  PollMode(InstrumentId,int) from U to A;
}
```

```
choice
    :       Choice 'at' roleName block ( 'or'  block )* -> ^(Choice roleName block+ );
```



Class diagrams


Graph
Pydot
Cluster
Dot
Subgraph
SGraph
Extensions
SSubgraph


Edge
Node
Pydot
SEdge
SNode
Extensions
Start
Message
Choice
Merge
Fork
Join
Rec
Interrupt
Delay
End