TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik » Institut für Software- und Multimediatechnik » Lehrstuhl Softwaretechnologie

Development of a User Interface for the Graphical Specification of Complex Rewrite Rules for the Reference Attribute Grammar Controlled Rewriting Library RACR

Adam Misiuda

Supervisor: Dipl.-Inf. Christoff Bürger

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Dresden, 24. Okt 2013

Outline

1. Project description
2. System architecture
3. Code generation and pattern matching
4. GUI
5. Conclusions

Outline

1. **Project description**
2. System architecture
3. Code generation and pattern matching
4. GUI
5. Conclusions

Project description

Main task:

- Development of a GUI for the specification of rewrite rules for RACR

Project description

Main task:

- Development of a GUI for the specification of rewrite rules for RACR

Realized task:

- Development of a GUI for the specification of the left hand's sided pattern matching for RACR

Project description

Main task:

- Development of a GUI for the specification of rewrite rules for RACR

Realized task:

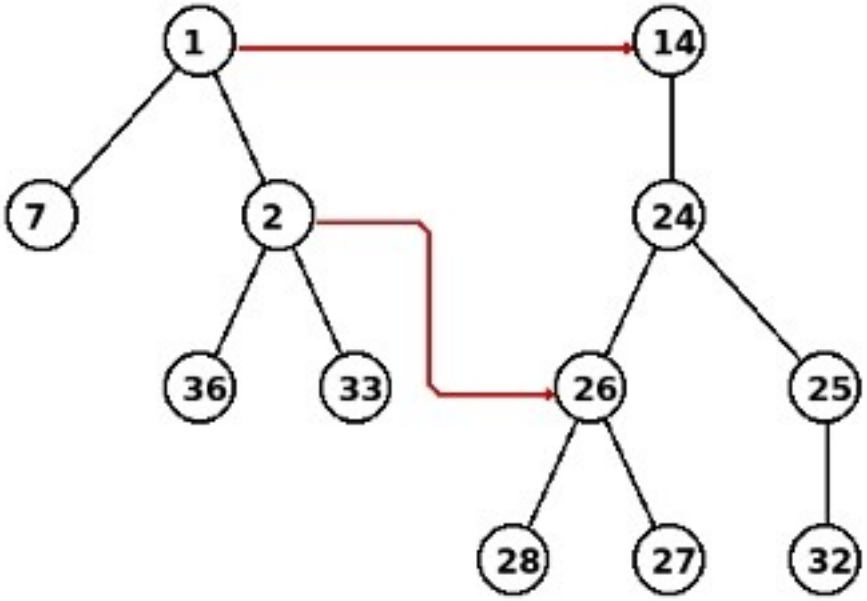
- Development of a GUI for the specification of the left hand's sided pattern matching for RACR

Technology:

- Scheme R6RS
- Racket 5.3.3
- RACR (Reference Attribute Grammar Controlled Rewriting) scheme library

Left hand's sided pattern

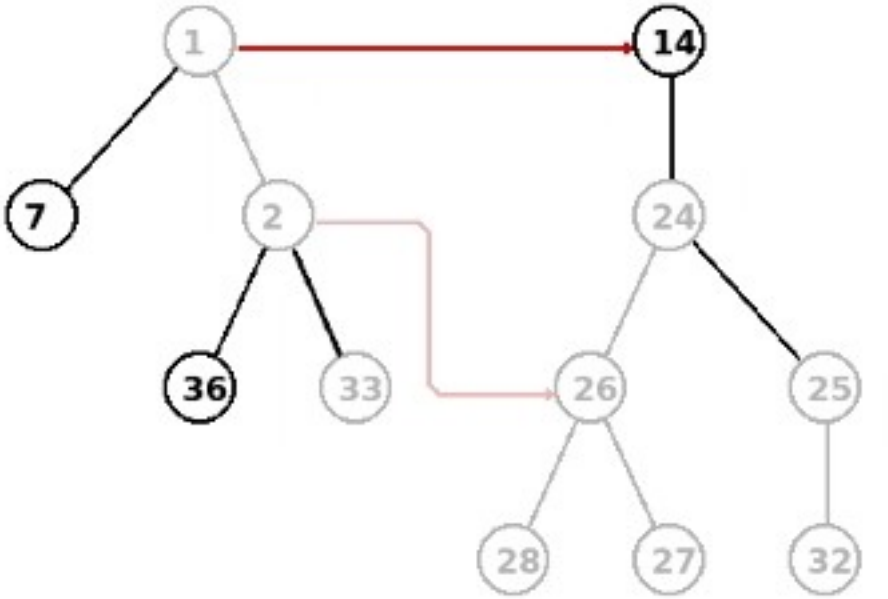
L-Side



R-Side

Left hand's sided pattern

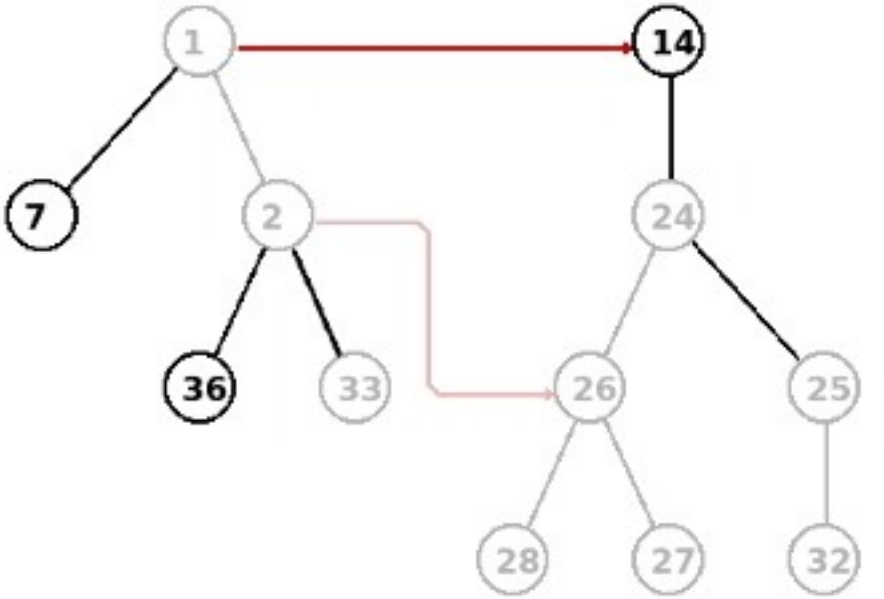
L-Side



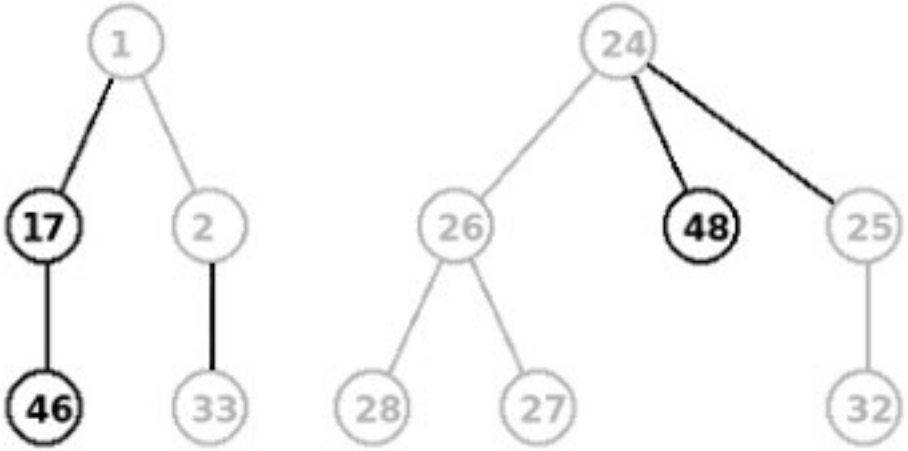
R-Side

Left hand's sided pattern

L-Side

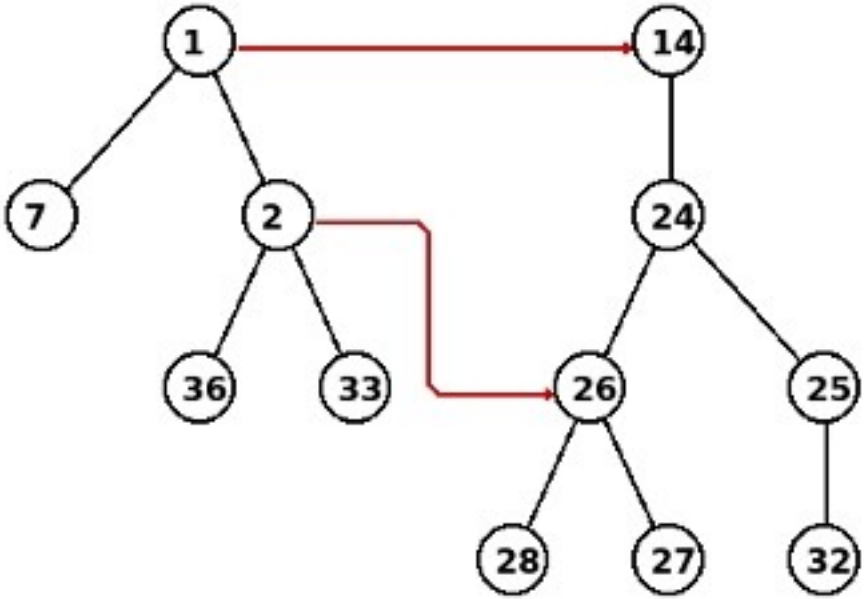


R-Side

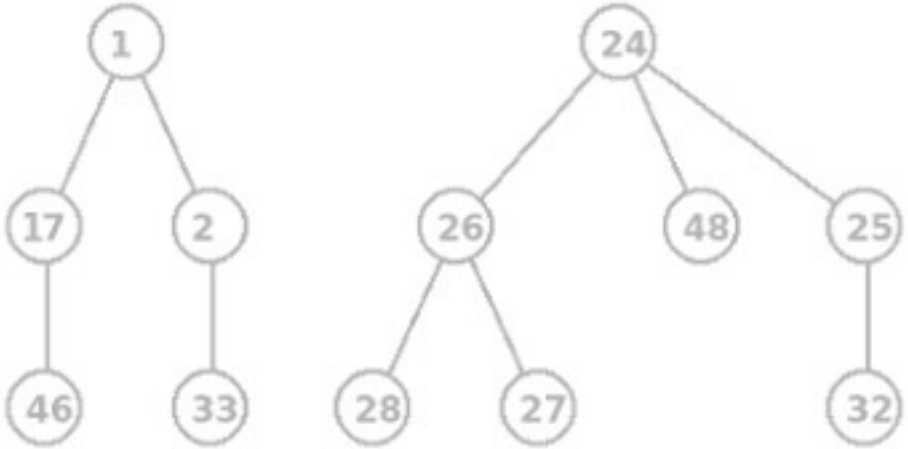


Left hand's sided pattern

L-Side



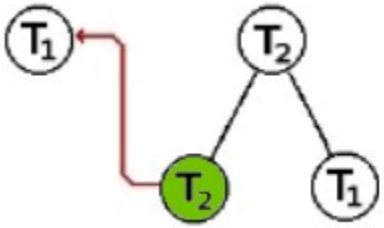
R-Side



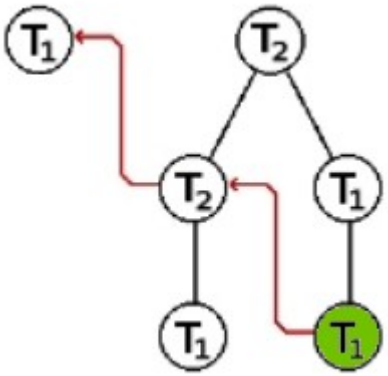
The project focuses only on the matching of the L-Side.

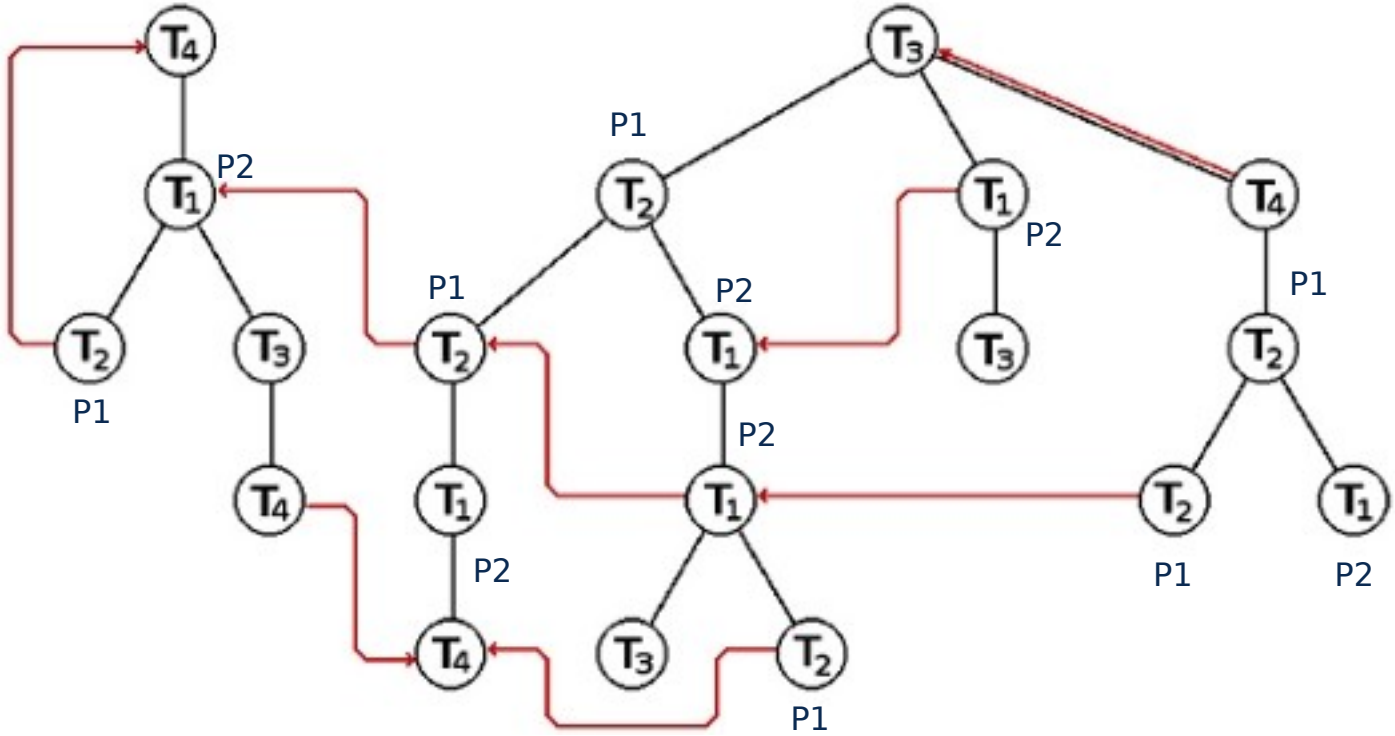
Pattern maching

Pattern 1 (P1)



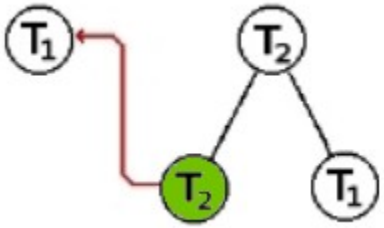
Pattern 2 (P2)



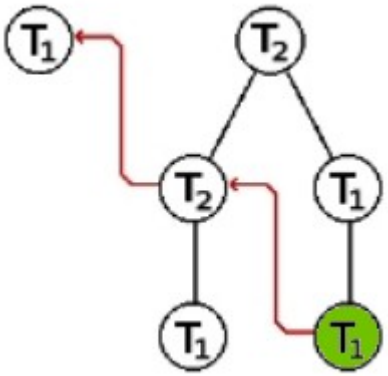


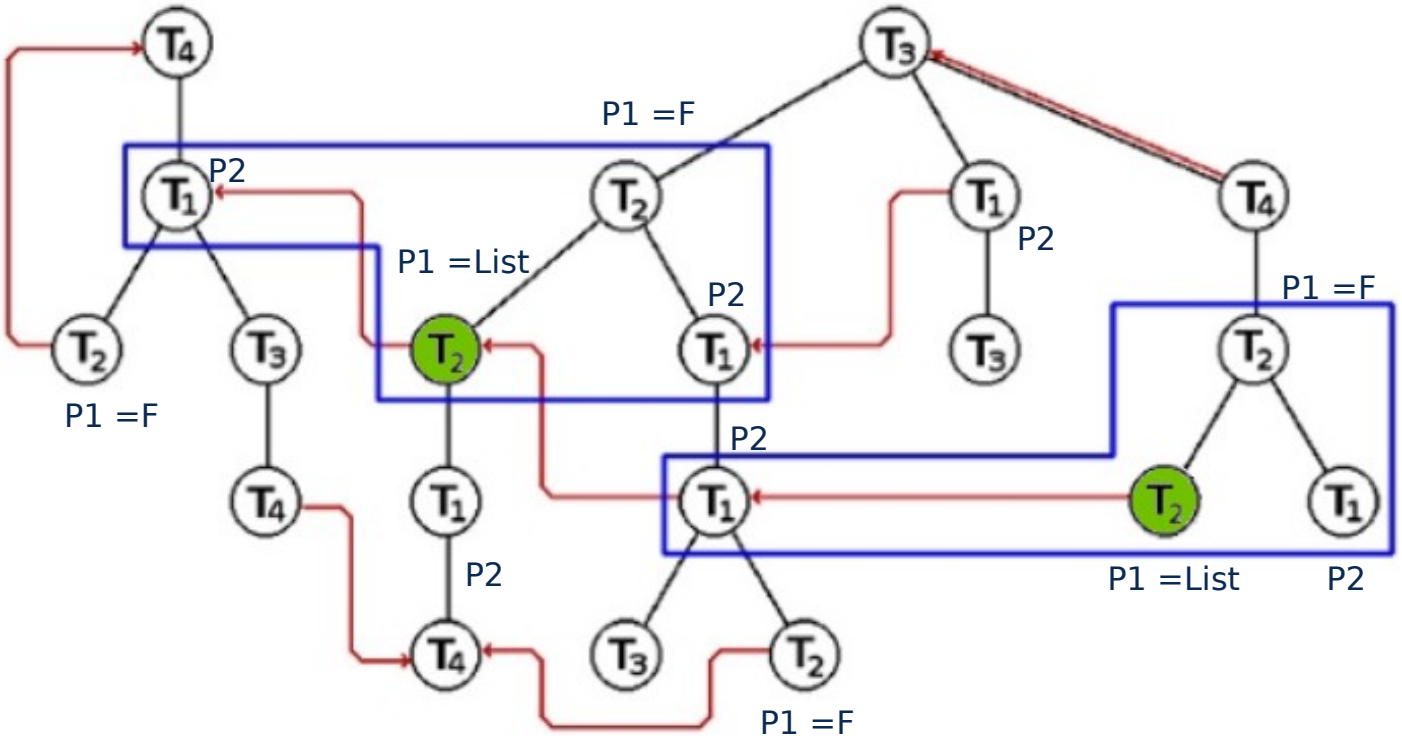
Pattern maching

Pattern 1 (P1)



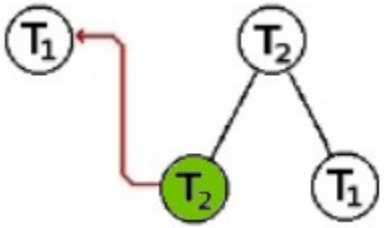
Pattern 2 (P2)



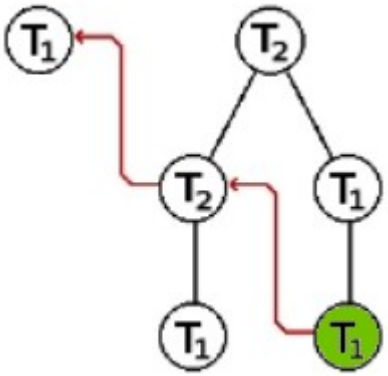


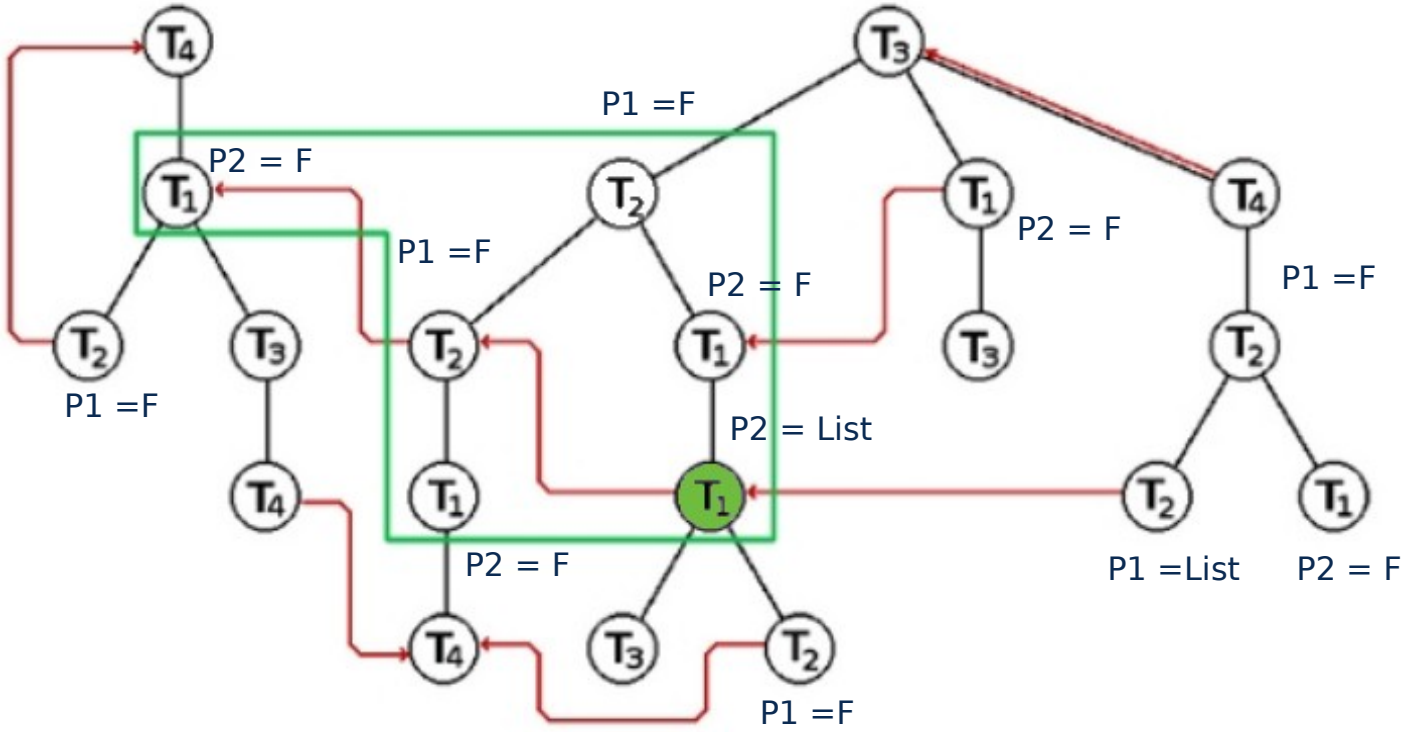
Pattern maching

Pattern 1 (P1)



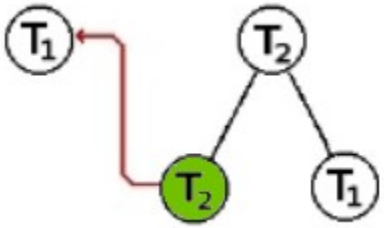
Pattern 2 (P2)



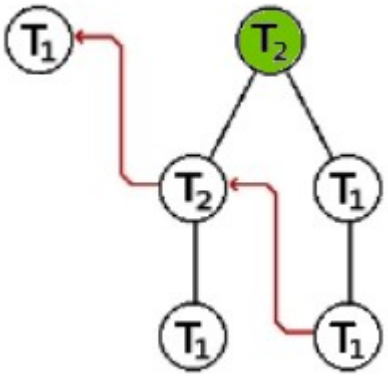


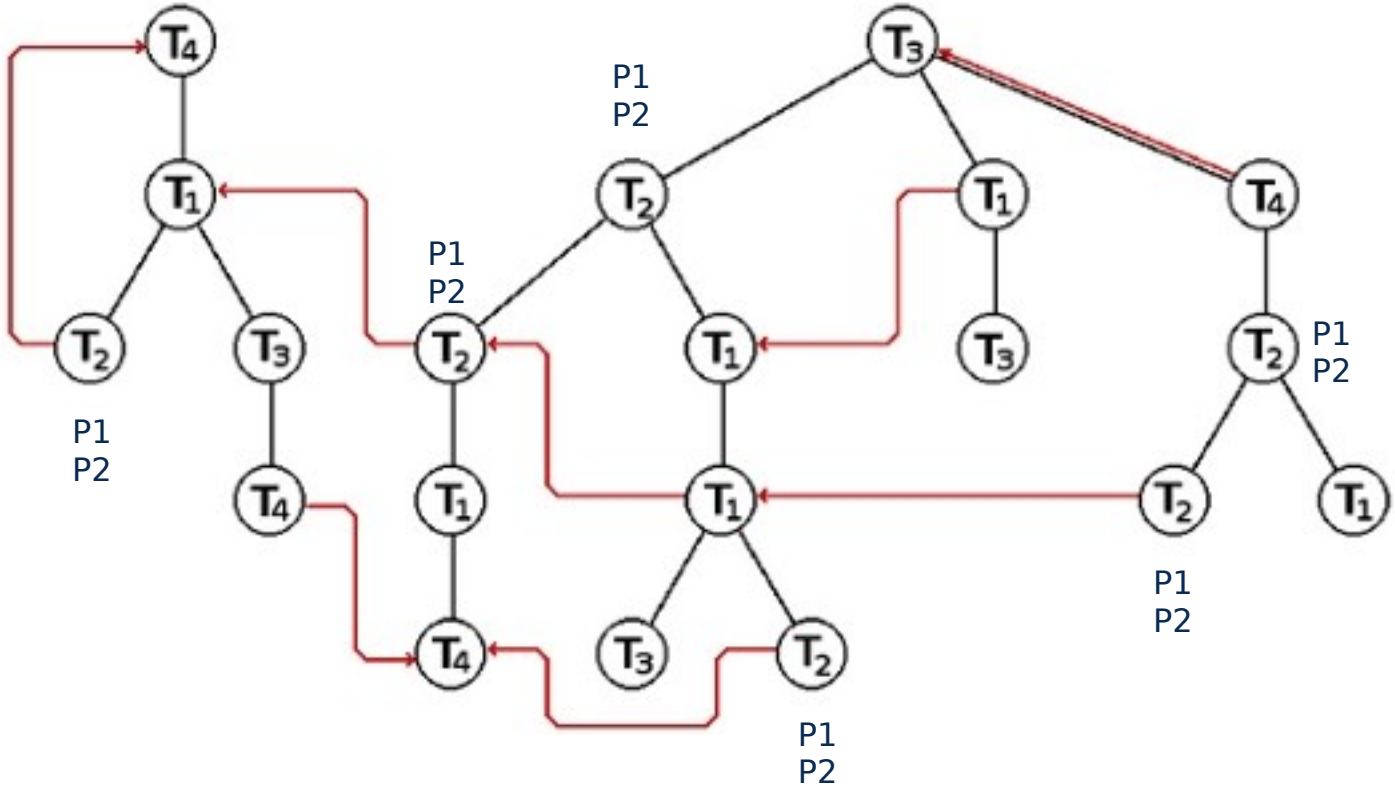
Pattern maching

Pattern 1 (P1)



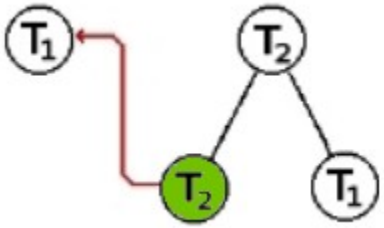
Pattern 2 (P2)



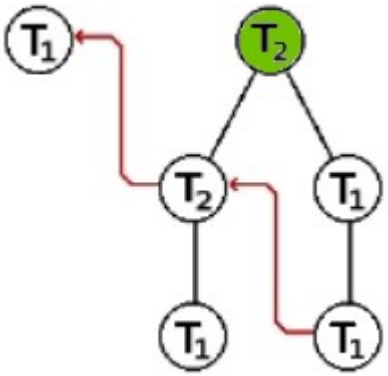


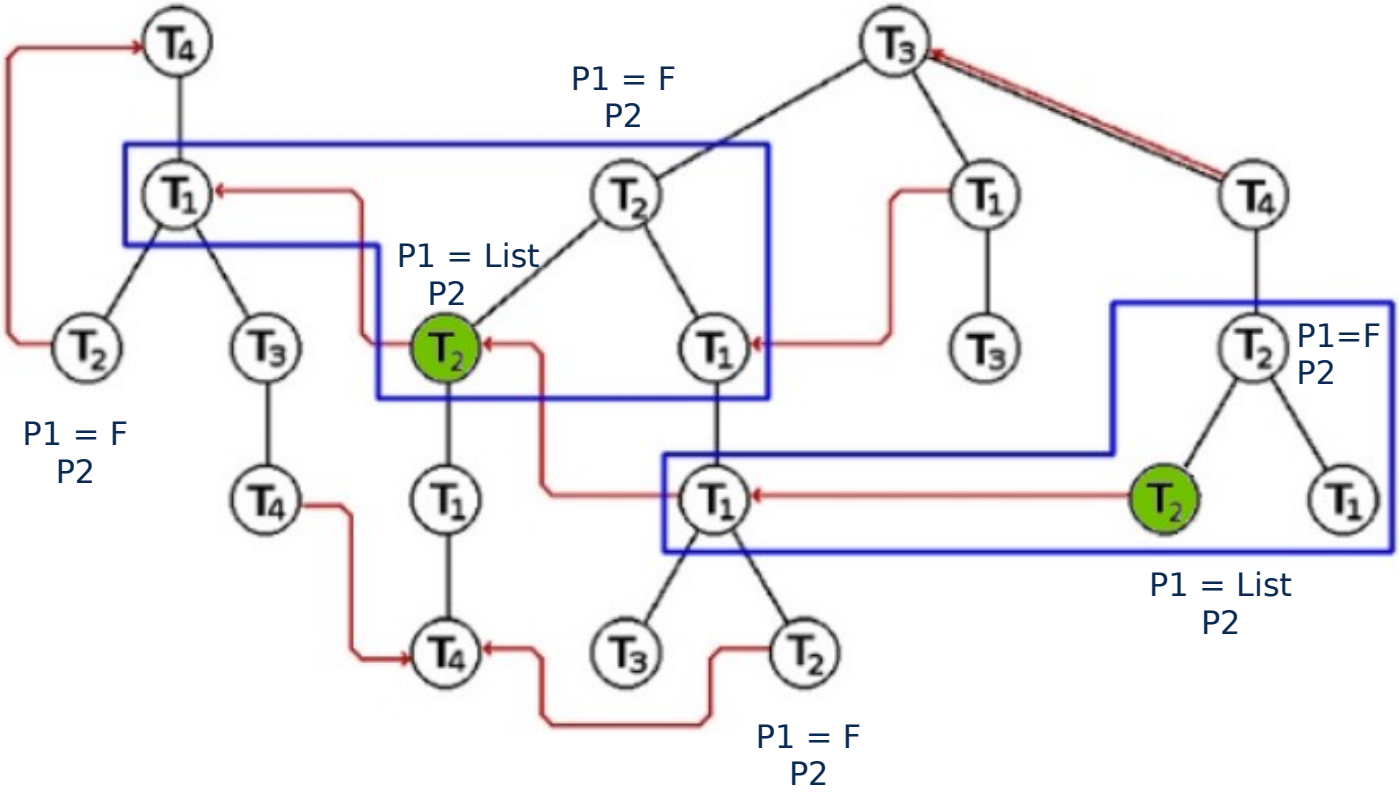
Pattern maching

Pattern 1 (P1)



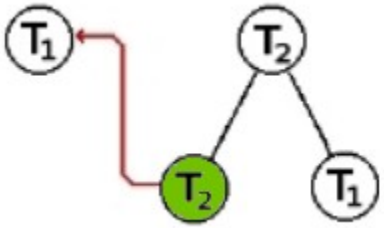
Pattern 2 (P2)



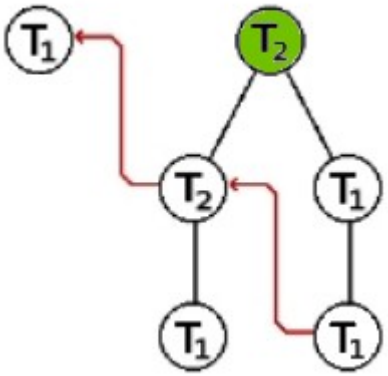


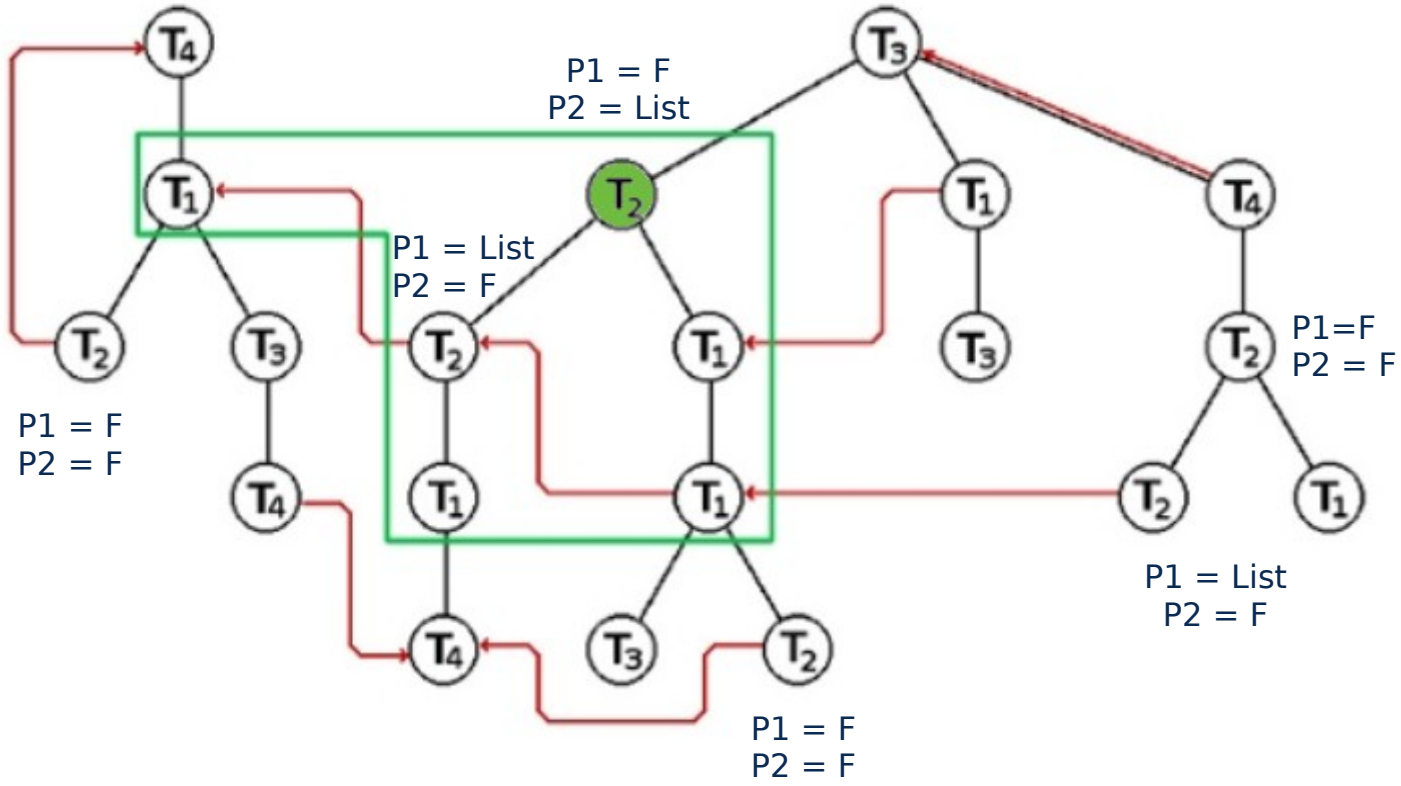
Pattern maching

Pattern 1 (P1)



Pattern 2 (P2)

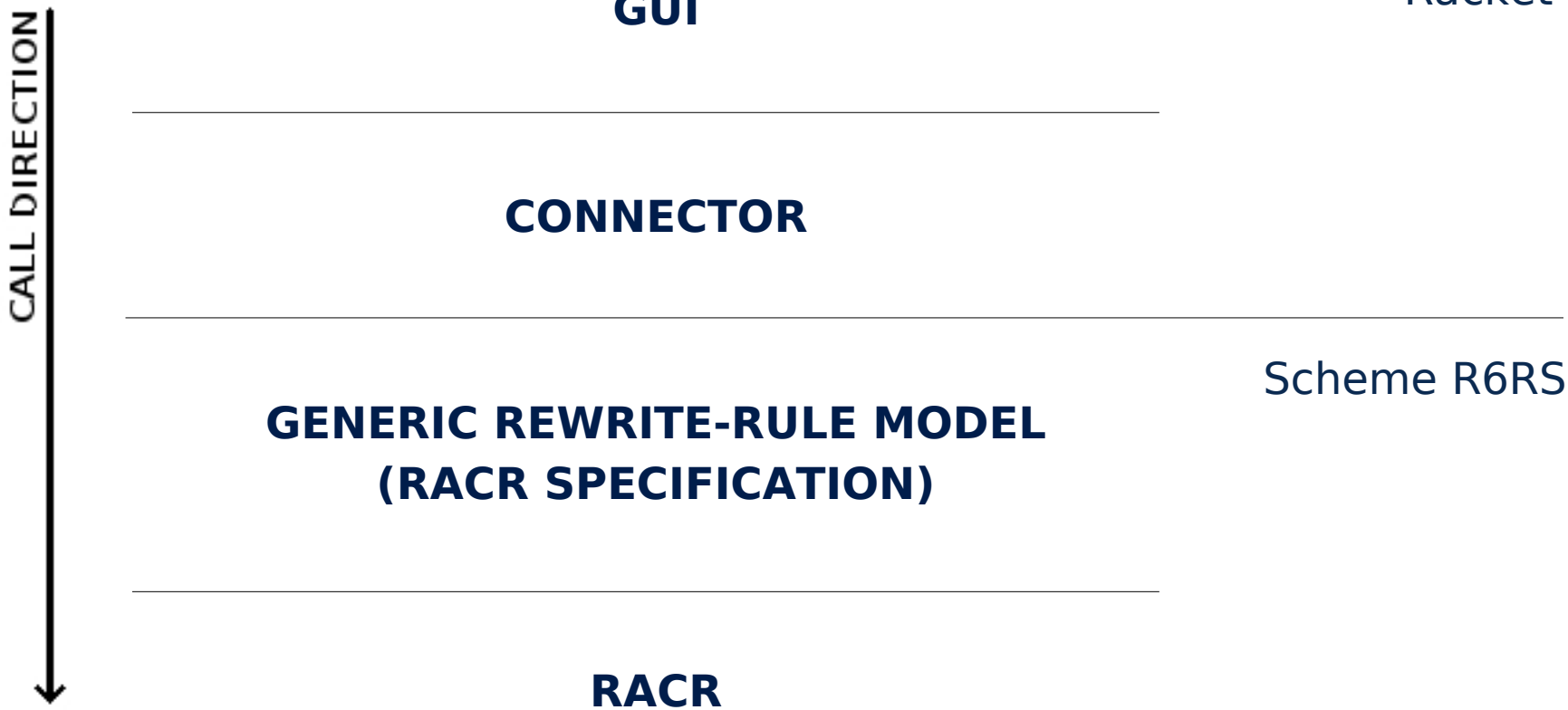




Outline

1. Project description
2. **System architecture**
3. Code generation and pattern matching
4. GUI
5. Conclusions

System architecture: layers



RACR description

- Scheme attribute grammar library

RACR description

- Scheme attribute grammar library
- Provides functions to:
  - specify abstract syntax tree (AST) schemes
  - specifies and queries AST's attribution
  - rewrites AST's attribution
  - Dynamic Attribute Dependency Analyses for efficient incremental attribute evaluation

System architecture: Semantics

- Well-formedness
  - Distinguished node test
  - Reachability test
  - Fragment types correctness

System architecture: Semantics

- Well-formedness
  - Distinguished node test
  - Reachability test
  - Fragment types correctness
- Layout
  - Node position
  - Reference track
  - Node visual representation

System architecture: Semantics

- Well-formedness
  - Distinguished node test
  - Reachability test
  - Fragment types correctness
- Code generation
  - mathing commands
- Layout
  - Node position
  - Reference track
  - Node visual representation

System architecture: Semantics

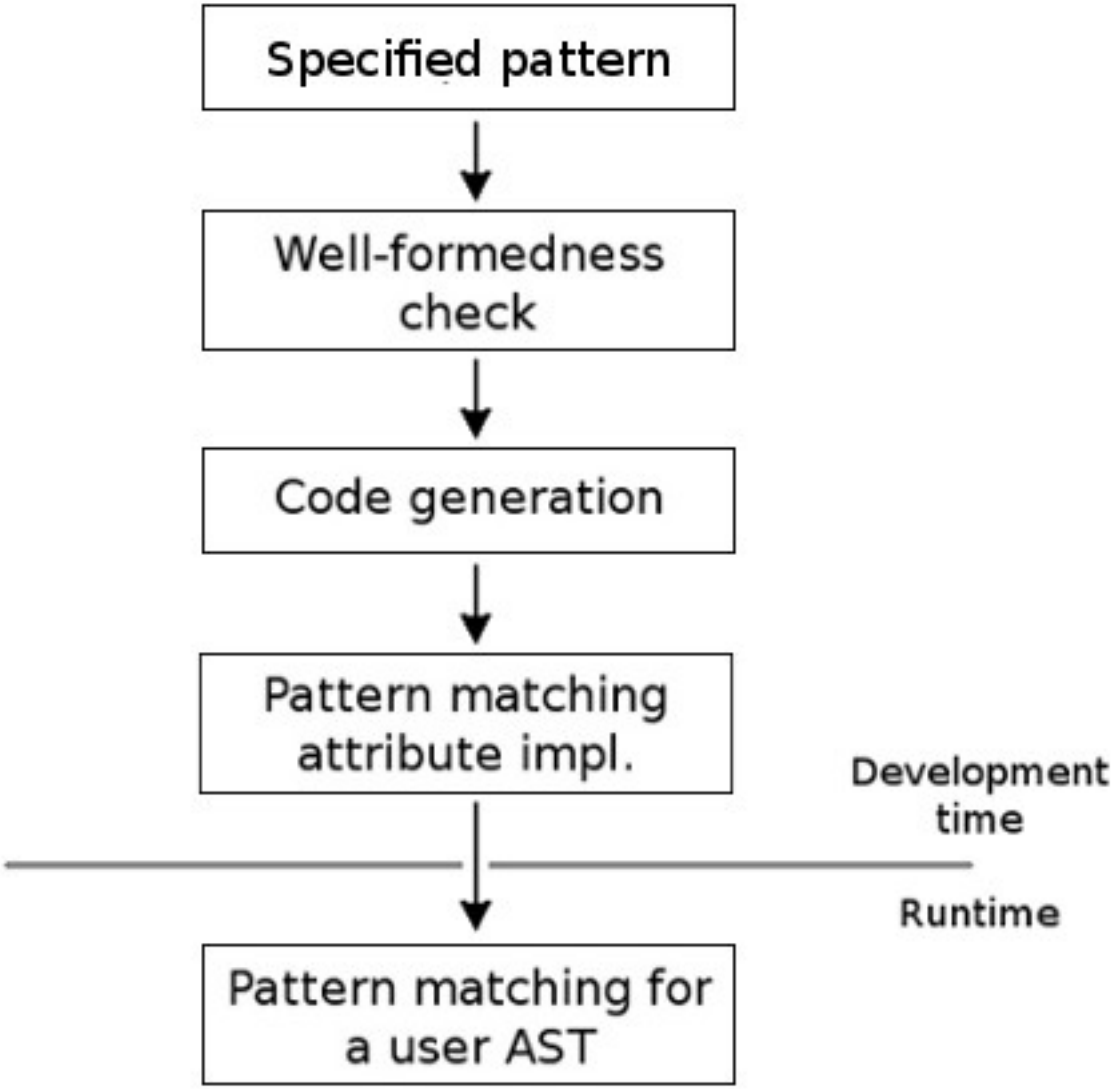
- Well-formedness
  - Distinguished node test
  - Reachability test
  - Fragment types correctness
- Code generation
  - mathing commands
- Layout
  - Node position
  - Reference track
  - Node visual representation

Semantics have been implemented using RACR

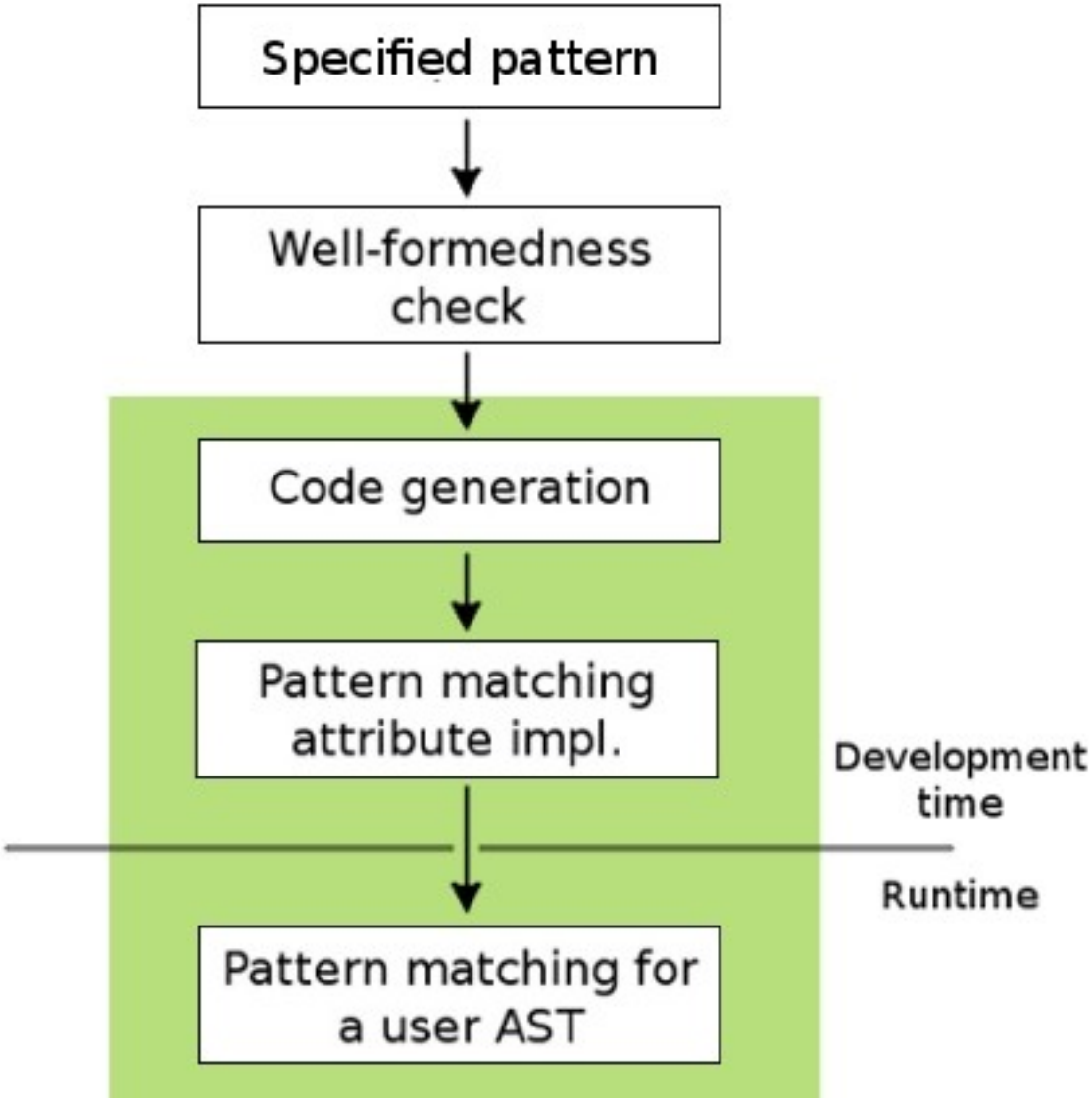
Outline

1. Project description
2. System architecture
3. **Code generation and pattern matching**
4. GUI
5. Conclusions

Workflow


Specified pattern
Well-formedness check
Code generation
Pattern matching attribute impl.
Development time
Runtime
Pattern matching for a user AST

Workflow


Specified pattern
Well-formedness check
Code generation
Pattern matching attribute impl.
Development time
Runtime
Pattern matching for a user AST

Code generation

- Performed on a well-formed specified pattern

Code generation

- Performed on a well-formed specified pattern
- Generates an implementation of the matching attribute for RACR
  - contains a list of the matching commands
  - specified only for a rule represented by the distinguished node's type

Matching commands

- Generated for:
  - pattern roots search
  - structure analysis
  - references consistency analysis

Matching commands

- Generated for:
  - pattern roots search
  - structure analysis
  - references consistency analysis
- No branching, loop or jump commands

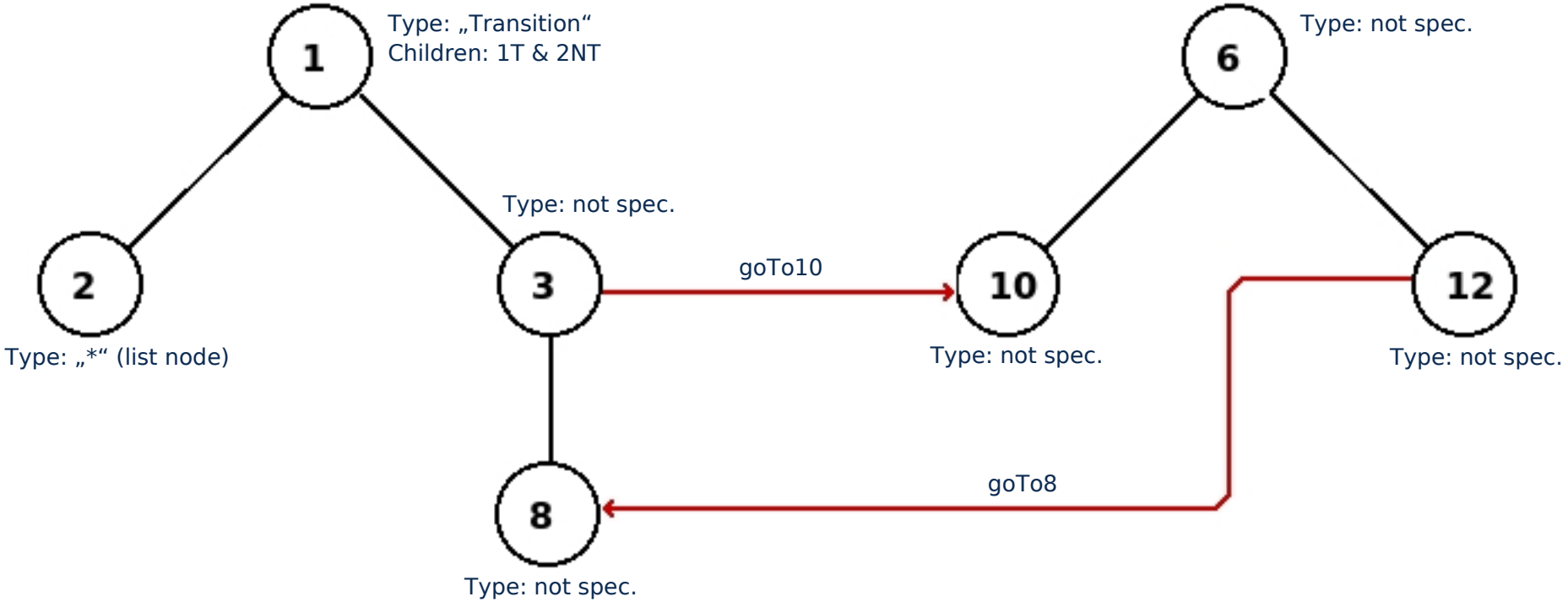
Matching commands

- Generated for:
  - pattern roots search
  - structure analysis
  - references consistency analysis
- No branching, loop or jump commands
- Executed command returns T or F value
  - aborts further commands execution on false

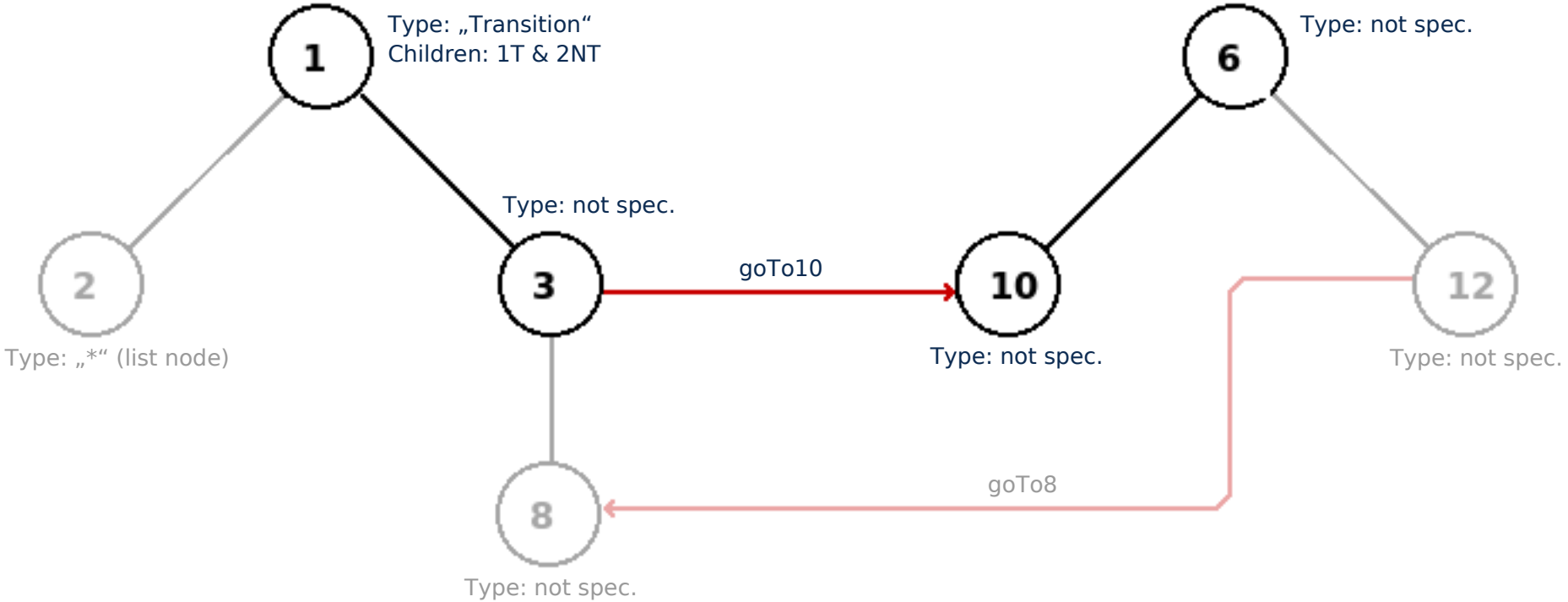
Pattern matching

- Currently availabe set of commands:
  - chType(str) — check node's Type
  - cR(id) — compare current node with one of the the nodes saved as roots
  - gCh(pos) — get child on given position
  - gR(id) — get root with specified number
  - numCh(num) — check number of node's children
  - p — go to the parent node
  - pCh(pos) — check node's position
  - r(str) — follow the specified reference
  - sName(str) — save node's name (binding)
  - sR(id) — save node as root

Matching commands generation



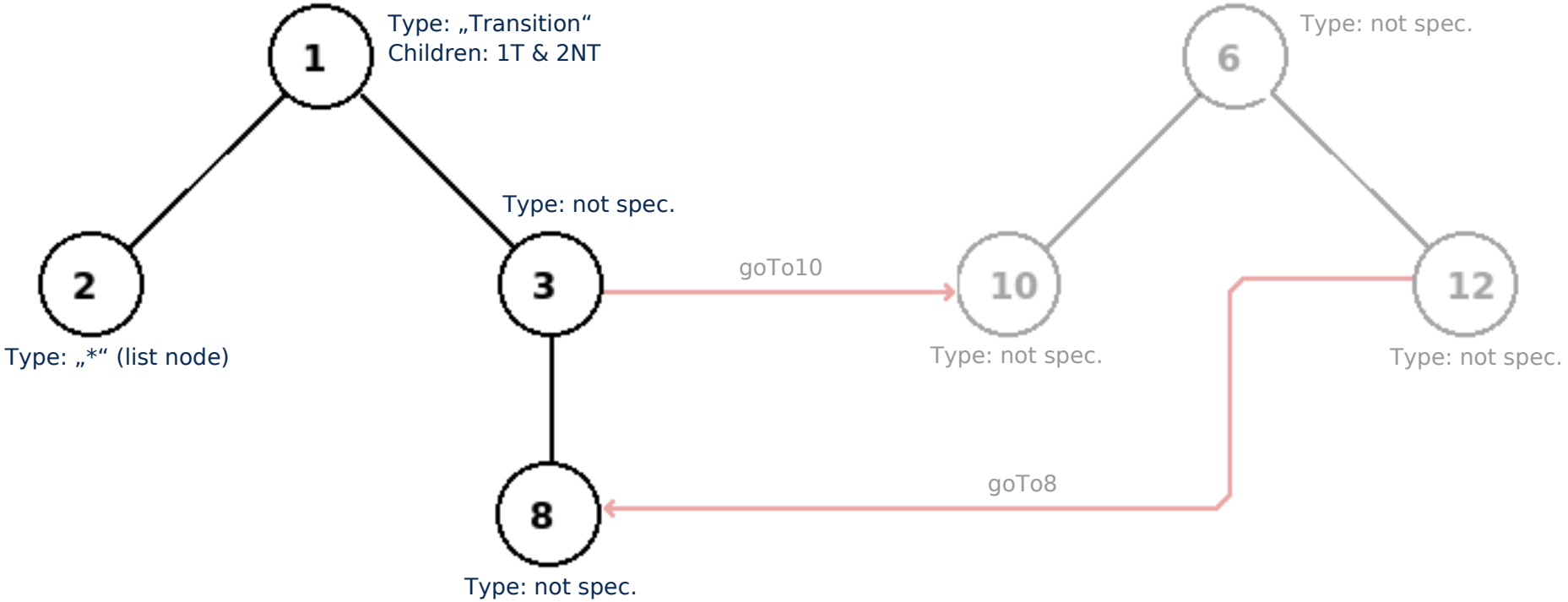
Matching commands generation



('sR 1)	save root
('gR 1)	get root
('numCh 3)	num. of children
('gCh 3)	get child

('r "goTo10")	follow the reference
('pCh -1)	children position
('p "")	go to parent node
('sR 6)	

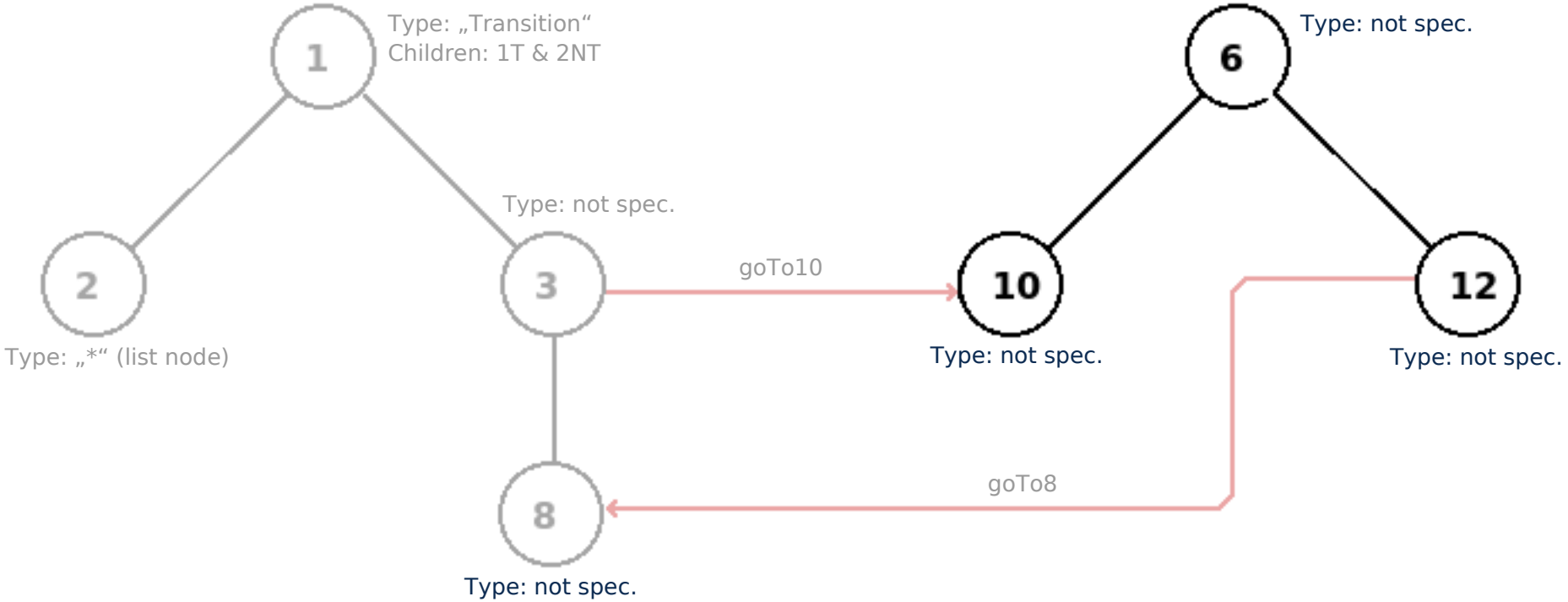
Matching commands generation



('gR 1)	get root
('chType "Transition")	check type
('numCh 3)	num. of children
('gCh 2)	get child
('chType "*")	

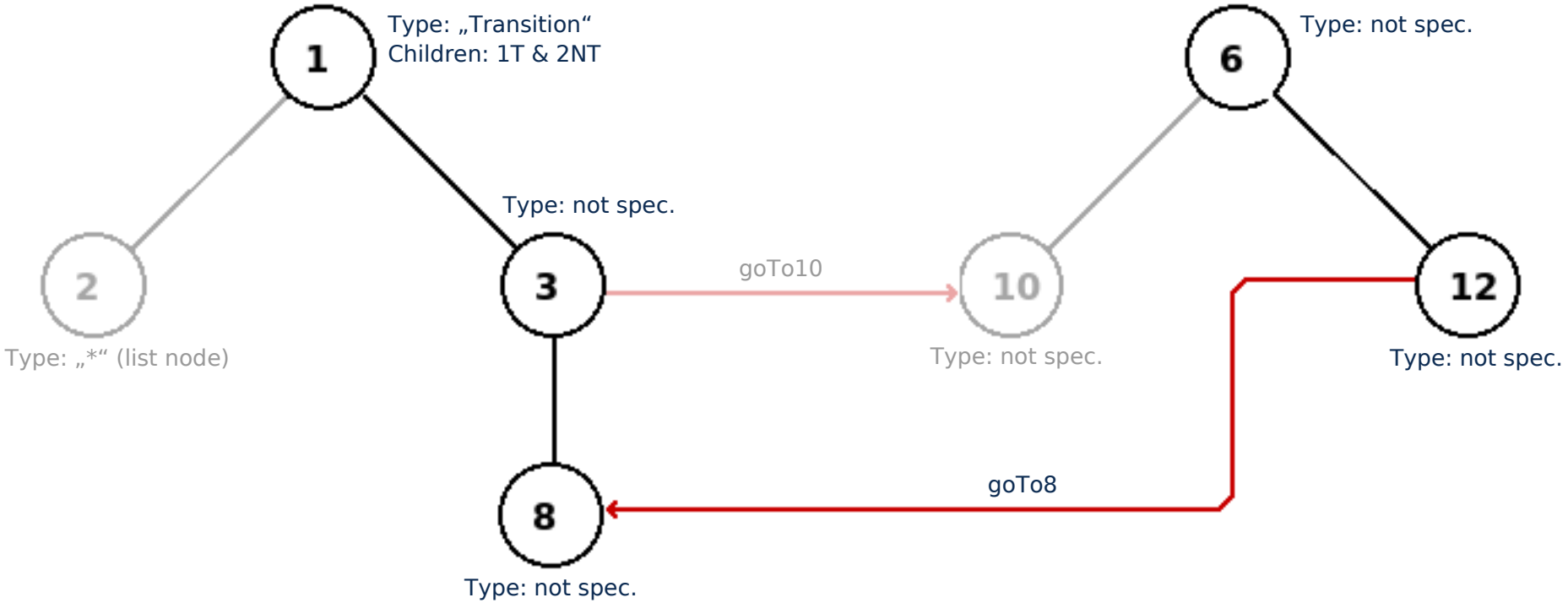
('p "")	go to parent node
('gCh 3)	
('numCh -1)	
('gCh -1)	

Matching commands generation



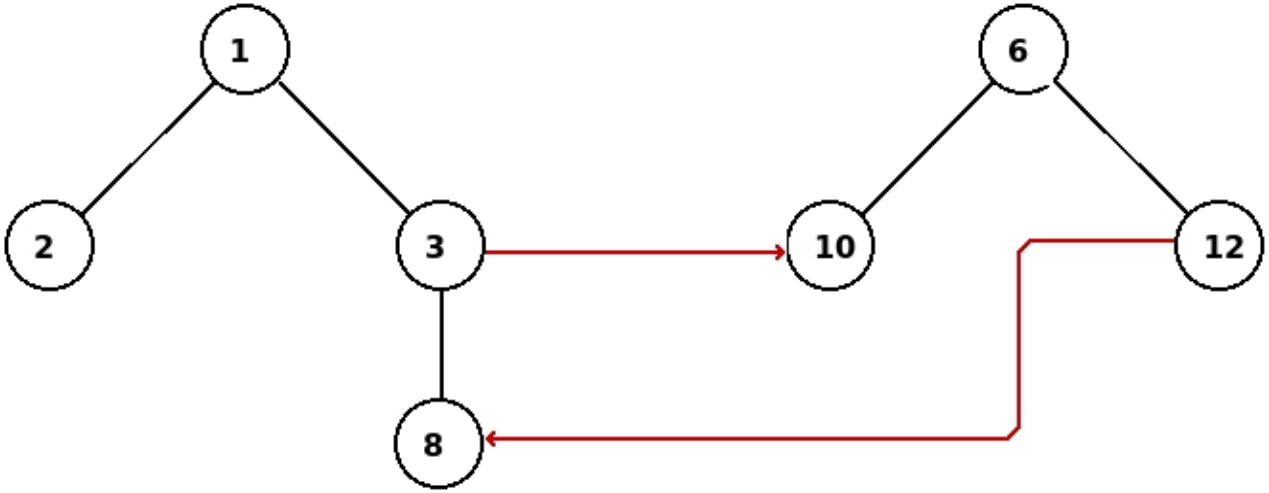
(cons 'gR 6)	get root
(cons 'numCh -2)	num. of children
(cons 'gCh -1)	get child
(cons 'p "")	go to parent node
(cons 'gCh -2)	

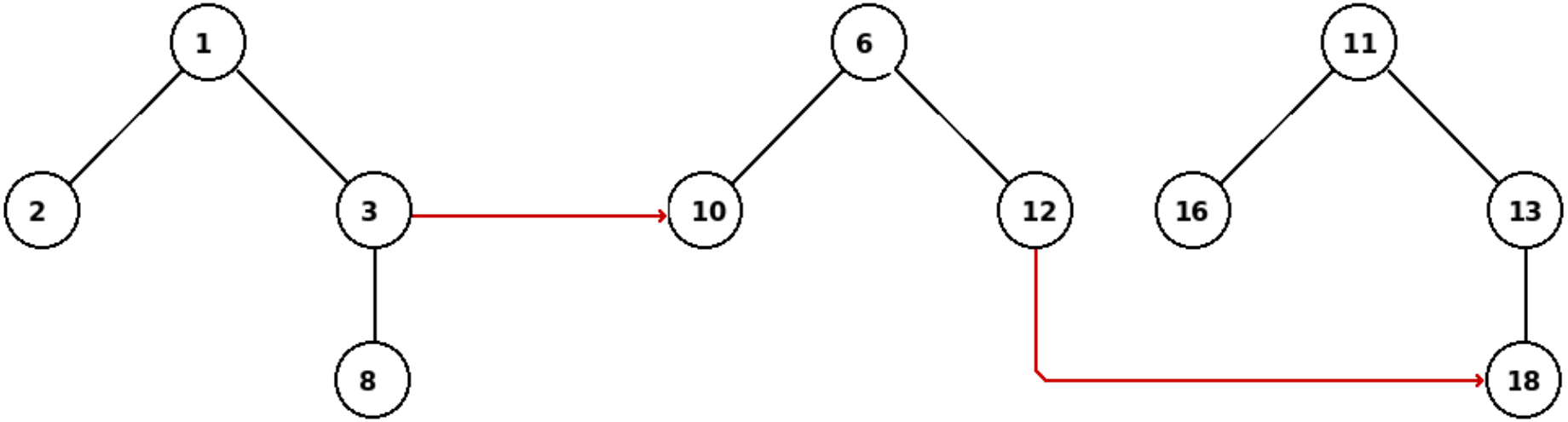
Matching commands generation



('gR 6)	get root	('p "")	go to parent node
('numCh -2)	num. of children	('pCh 3)	
('gCh -2)	get child	('p "")	
('r "goTo8")	follow the reference	('cR 1)	compare root
('pCh -1)	child position		

Matching commands generation





Abstract matching

- Uses an interpreter of the matching commands
  - executes each command in the given order
  - returns false on execution abort
  - returns bidings list on success

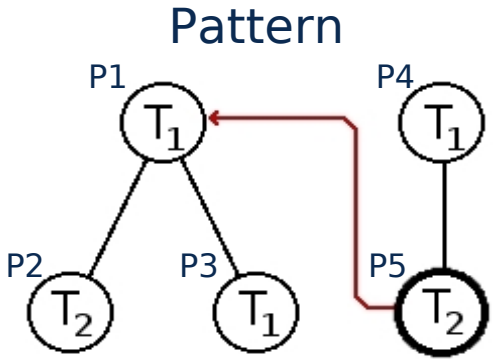
Abstract matching

- Uses an interpreter of the matching commands
  - executes each command in the given order
  - returns false on execution abort
  - returns bidings list on success
- O(n) complexity
  - due to no branching, loop or jump commands
  - execution time depends only on the commands list size

Abstract matching

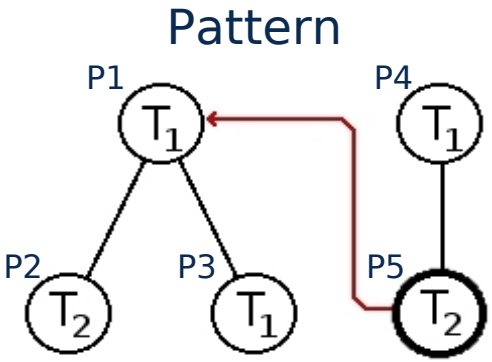
- Uses an interpreter of the matching commands
  - executes each command in the given order
  - returns false on execution abort
  - returns bidings list on success
- O(n) complexity
  - due to no branching, loop or jump commands
  - execution time depends only on the commands list size
- Easy commands list extension
  - new types of commands require only interpreter extension
  - backward compatibility

Pattern matching example

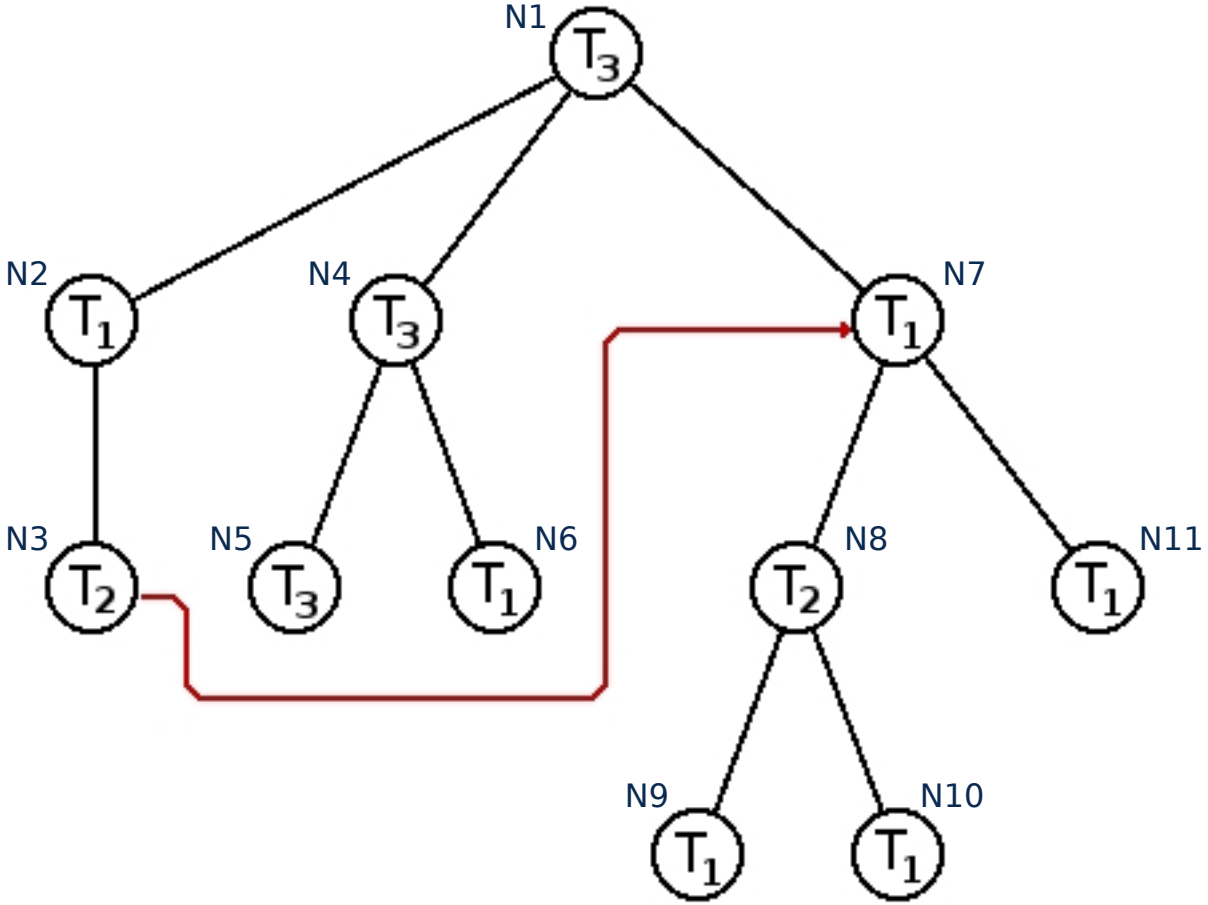


Where:
P2 name = „node2“
P4 name = „root2“

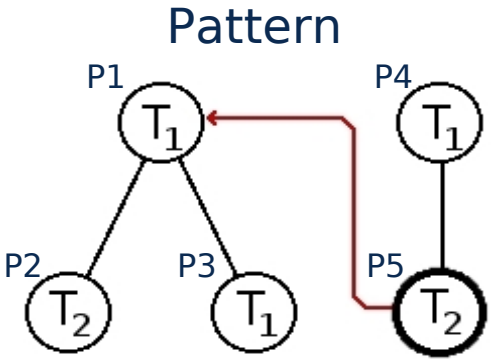
Pattern matching example



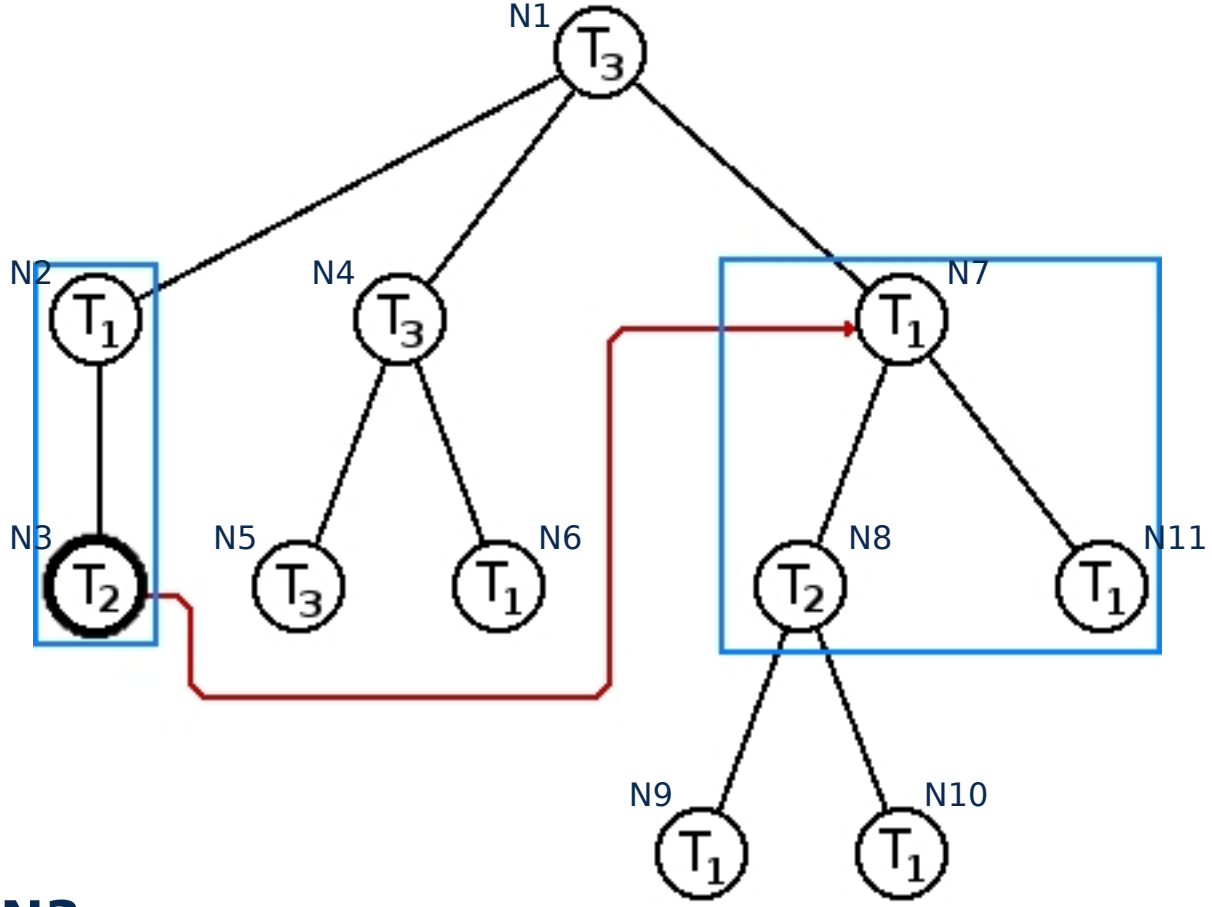
Where:
P2 name = „node2“
P4 name = „root2“



Pattern matching example

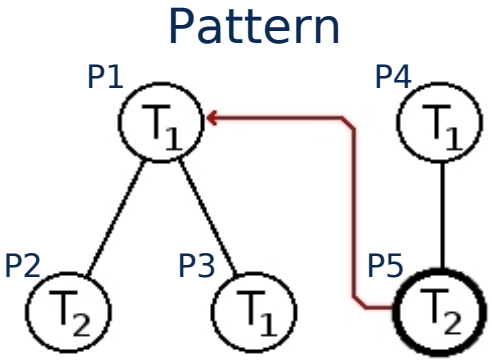


Where:
P2 name = „node2“
P4 name = „root2“

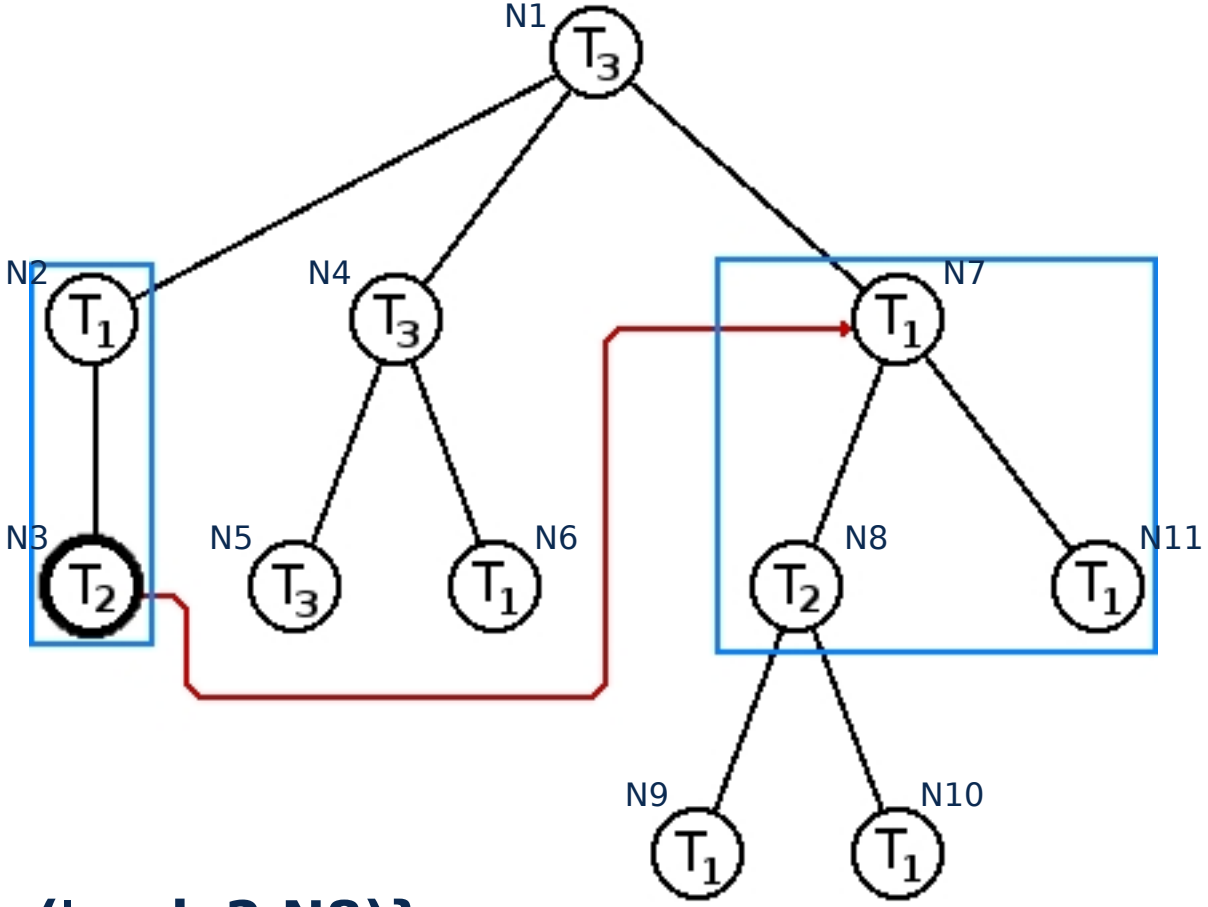


Matching found for N3

Pattern matching example



Where:
P2 name = „node2“
P4 name = „root2“



Result: {('root2 N2), ('node2 N8)}

Outline

1. Project description
2. System architecture
3. Code generation and pattern matching
4. **GUI**
5. Conclusions

DEMO

Outline

1. Project description
2. System architecture
3. Code generation and pattern matching
4. GUI
5. **Conclusions**

Conclusions

- Easy to use GUI

Conclusions

- Easy to use GUI
- Integration with the current RACR version

Conclusions

- Easy to use GUI
- Integration with the current RACR version
- Easy generation of the pattern matching attributes thanks to the GUI

Conclusions

- Easy to use GUI
- Integration with the current RACR version
- Easy generation of the pattern matching attributes thanks to the GUI
- Commands list provides easy extendability

Conclusions

- Easy to use GUI
- Integration with the current RACR version
- Easy generation of the pattern matching attributes thanks to the GUI
- Commands list provides easy extendability
- Next step: specification of rewrite rules for RACR

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik » Institut für Software- und Multimediatechnik » Lehrstuhl Softwaretechnologie

Development of a User Interface for the Graphical Specification of Complex Rewrite Rules for the Reference Attribute Grammar Controlled Rewriting Library RACR

Adam Misiuda

Supervisor: Dipl.-Inf. Christoff Bürger

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Dresden, 24. Okt 2013