universitätbonn ROOTS

Conditional Transformations

Fabian Noth

noth@cs.uni-bonn.de

What are Conditional Transformations?

- Combining analysis and transformations on programs
  - (nearly) every transformation requires an analysis
- Based on logic
  - Program as logic factbase
- Uniformity
  - Independecy from underlying system (e.g. java source code)
- Complex sequences
  - Combination of simple CT

Program as logic factbase

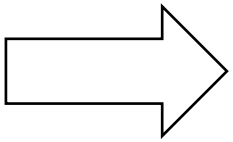
Source code

```
class A {
   public int a;
   private double b;
   float c;

   void m(int i)
   {
        m(i);
   }

}

class B {
   public int j;
   private int k;
   public boolean b;
}
```



Factbase

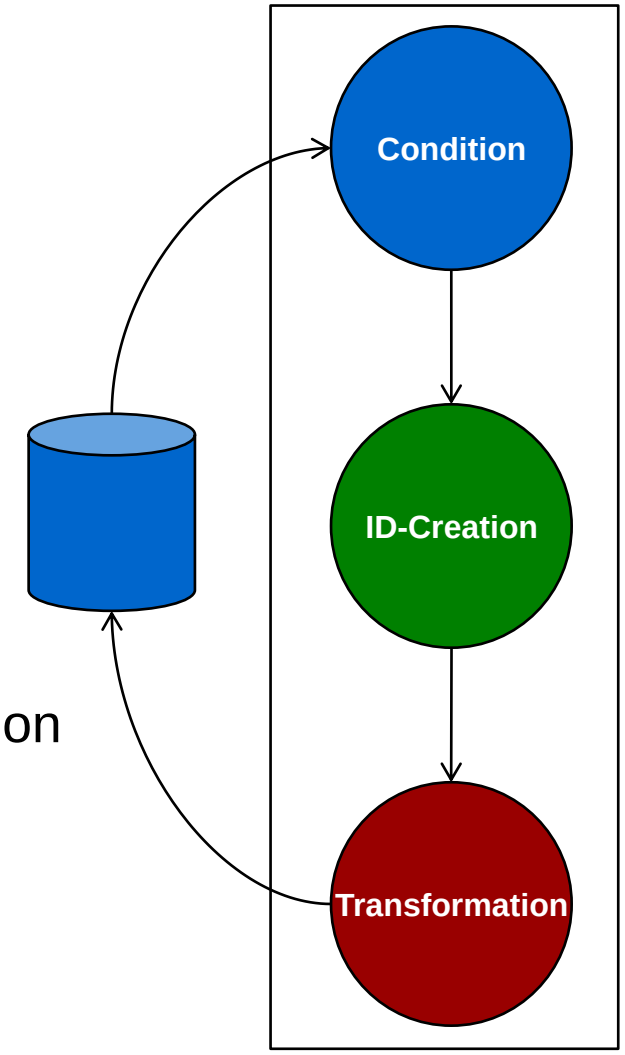
```
classT(1,0,'A',[2,3,4,5])
fieldT(2,1,int,'a')
modifierT(2, ,public')
fieldT(3,1,double,'b')
modifierT(3, 'private')
fieldT(4,1,float,'c')
modifierT(4, 'package')

methodT(5,1,'m',[6],void, 7)
paramT(6,5,int,'i')
blockT(7,5)
...

classT(12,0,'B',[13,14,15])
fieldT(13,12,int,'j')
modifierT(13, 'public')
fieldT(14,12,int,'k')
modifierT(14, 'private')
fieldT(15,12,boolean,'b')
modifierT(15, 'public')
```

Conditional Transformation

- Program as logic factbase
- Evaluate if the condition is true
  - Analysing the factbase
- Propagation of valid substitutions (IDs)
- Creation of missing IDs
- Propagation of all required IDs for Transformation
- Changing the factbase



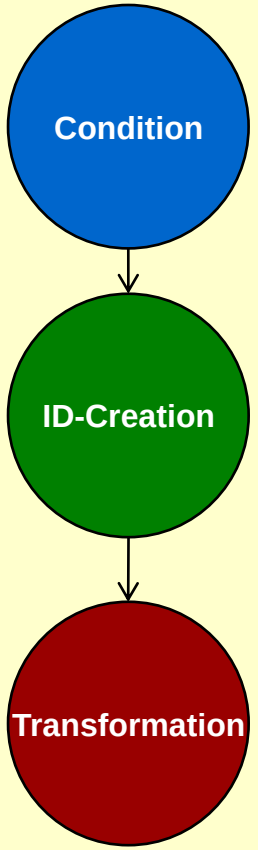
Example: copyFields

```
1  ct( copyField( SrcClass, SrcField, Ftype, Fname, TargetClass, TargetField),

2     condition( (
3       fieldT(SrcField, SrcClass, Ftype, Fname, _),
4       not( fieldT(_,TargetClass, _, Fname, _) ),
5       modifierT(SrcField, Mod)
6     ) ),

7     idcreation( (
8       new_node_id(TargetField)
9      ) ),

10    transformation( (
11      add(fieldT(TargetField, TargetClass, Ftype, Fname, null)),
12      add(modifierT(TargetField, Mod)),
13      add_to_class(TargetClass, TargetField)
14    ))
15 ).
```



Evaluating the condition

Condition

Input: `SrcClass = 1, TargetClass = 12`

```
condition( (
  fieldT(SrcField,SrcClass, Ftype, Fname, _),
  not( fieldT(_,TargetClass, _, Fname, _) ),
  modifierT(SrcField, Mod)
) ),
```

SrcClass	Target Class	SrcField	Ftype	Fname	Mod
1	12	2	int	a	public
1	12	4	float	c	package

Substitution Set

Factbase

```
classT(1,0,'A',[2,3,4,5])
fieldT(2,1,int,'a')
modifierT(2, ,public')
fieldT(3,1,double,'b')
modifierT(3, 'private')
fieldT(4,1,float,'c')
modifierT(4, 'package')

...

classT(12,0,'B',[13,14,15])
fieldT(13,12,int,'j')
modifierT(13, 'public')
fieldT(14,12,int,'k')
modifierT(14, 'private')
fieldT(15,12,boolean,'b')
modifierT(15, 'public')
```

ID-Creation

- No unbound variables are allowed in the transformation
- Variables which are not bound in the condition have to be bound in the ID-Creation
- **`new_node_id(Var)`**-Command
  - Creates new, unique ID
  - Binds variable Var to this ID
  - For each substitution
- **`skip`**-Command
  - If all variables are bound in the condition

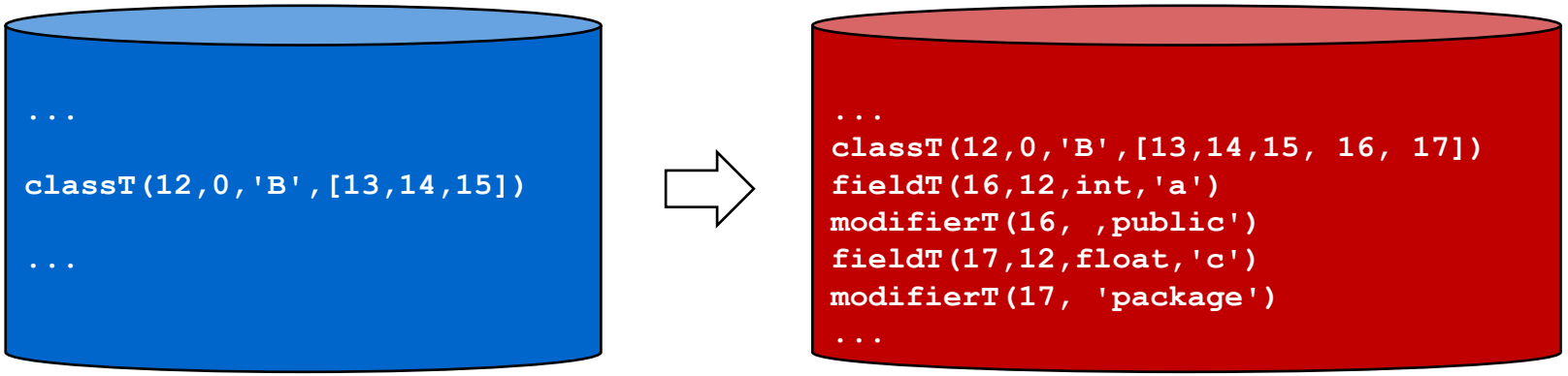
```
idcreation( (
  new_node_id(TargetField)
) )
```

TargetField	...
16	...
17	...

Transformation

- Transformation = Combination of:
  - add
  - delete
  - replace
  - Prolog predicates (e.g **`add_to_class`**)

```
transformation( (
  add(fieldT(TargetField, TargetClass, Ftype, Fname, null)),
  add(modifierT(TargetField, Mod)),
  add_to_class(TargetClass, TargetField)
))
```



Blackbox

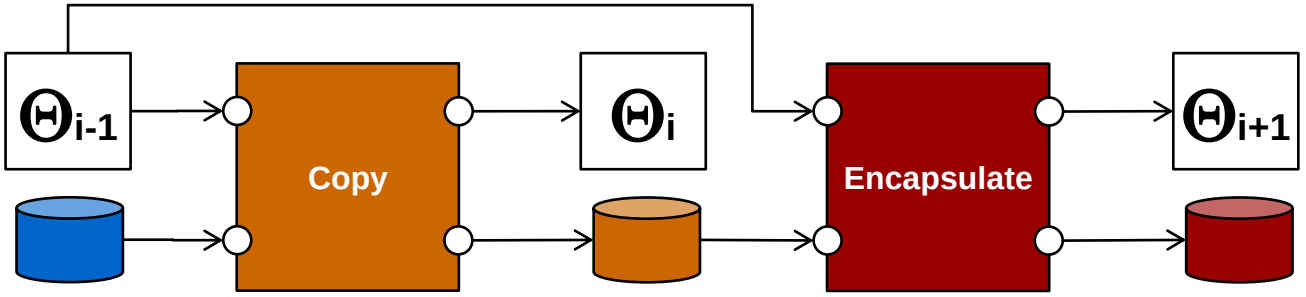
```
copyField( SrcClass, SrcField, Ftype, Fname, TargetClass, TargetField)
```

Blackbox

Sequences

```
ctseq(
  copyAndEncapsulateFields_Or(SrcClass,SrcField,TargetClass,TargetField),

  orseq(
   ct(copyField(SourceClass,SourceField,FType,Fname,TargetClass,TargetField)),
   ct(encapsulateField(SourceField))
  )
).
```

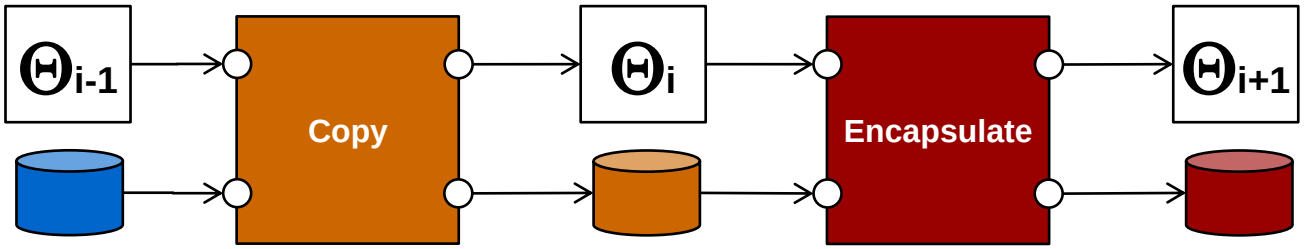


- Copy all possible fields
- Encapsulate all possible fields
  - No matter if it is a copied field or not

Sequences

```
ctseq(
  copyAndEncapsulateFields_Prop(SrcClass,SrcField,TargetClass,TargetField),

  propseq(
   ct(copyField(SourceClass,SourceField,FType,Fname,TargetClass,TargetField)),
   ct(encapsulateField(SourceField))
  )
).
```



- Copy all fields
- Encapsulate only copied fields

CTs in Action

- CT-Example in Eclipse
- Copy fields
- Example for use of sequences