

Turtle: A Constraint Imperative Programming Language

Martin Grabmüller and Petra Hofstedt
{magr,ph}@cs.tu-berlin.de
Fakultät IV – Elektrotechnik und Informatik
Technische Universität Berlin
Berlin, Germany



Constraint Imperative Programming

- Imperative
 - Stateful, time-dependent
 - Useful for modelling interactive systems
 - Side-effecting I/O
- Constraint-based
 - Declarative
 - High-level problem specifications
 - Separate solving algorithm
- Constraint imperative programming combines advantages
 - Suitable algorithms for varying requirements
 - Clean specification and efficient execution



Turtle: Language Design

- Imperative base language
 - Procedural language (procedures, statements, loops, assignments)
 - Functional extensions (HOF, algebraic data types, polymorphism)
- Constraint programming extensions
 - Constrainable variables
 - Constraint statements
 - User-defined constraints
 - Constraint solvers

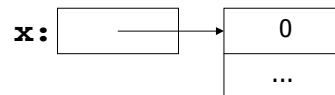


Constraint Extensions

- Constrainable variables

```
var x: !int := var 0;
```

- Constraint type \uparrow
- Variable object \uparrow



- Constraint statements

```
require 0 <= x and x <= 0 in
```

```
  y := x;
```

```
end;
```

- Constraint conjunction
- Body statements
- Constraint hierarchies



Constraint Extensions

- User-defined constraints

- Abstract over constraint conjunctions

```
constraint domain(x: !int, min: int, max: int)
  require min <= x and x <= max;
end;
require domain(x, 2, 9) and domain(y, 4, 6);
```

- Constraint solvers

- Maintain conjunction of active constraints
- Check satisfiability and calculate variable assignments



Example

```
1 constraint all_different (l: list of !int)
2   while (tl l <> null) do
3     var ll: list of !int := tl l;
4     while (ll <> null) do
5       require hd l <> hd ll;
6       ll := tl ll;
7     end;
8     l := tl l;
9   end;
10 end;
11 var a: !int := var 0, b: !int := var 0,
12     c: !int := var 0;
12 require all_different ([a, b, c]) in ... end;
```



Constraint Analysis

```
var y: int := 4;  
var x: !int := var 0;  
require 10*x + 10 > 3*y - 1;
```

- Constant expression:
`-11 + 3*y`
- Constrainable variable with coefficient:
`10*x`



Code Generation

```
10*x > -10 + 3*y - 1
```

Constant expression: `-11 + 3*y`

Constrainable variable and coefficient: `10*x`

```
1 push-constant 0 // constraint strength '0'  
2 push-constant 3 // constraint kind '>'  
3 push-constant 1 // number of constrainable variables  
4 push-variable y // calculate the constant term...  
5 load-constant 3  
6 mul  
7 push  
8 load-constant -11  
9 add  
10 push  
11 push-variable x // load constrainable variable object  
12 push-constant 10  
13 add-constraint // add the constraint to the store
```



Conclusion

- Turtle: constraint imperative programming language
 - Imperative base language
 - Functional language features
 - Constraint extensions
- Implementation
 - Standard compiler and constraint analysis
 - Run-time system with garbage collection and OS interface
 - Library with utility modules
- Future work
 - More and better constraint solvers
 - Language support for nondeterministic computation

