NUMERICAL TRANSITION SYSTEMS COMPETITION

Software verification ...

Software verification …

```
int syracuse(int n) {
  if (even(n))
    return n/2;
  return 3*n+1;
}

main() {
  int n = random();
  while (n != 1)
     n = syracuse(n);
}
```

Software verification …

```
int syracuse(int n) {
  if (even(n))
    return n/2;
  return 3*n+1;
}

main() {
  int n = random();
  while (n != 1)
     n = syracuse(n);
}
```

n=10, 5, 16, 8, 4, 2, 1

Hard, in theory …

```
int syracuse(int n) {
  if (even(n))
    return n/2;
  return 3*n+1;
}

main() {
  int n = random();
  while (n != 1)
     n = syracuse(n);
}
```

Hard, in theory …

```
int syracuse(int n) {
  if (even(n))
    return n/2;
  return 3*n+1;
}

main() {
  int n = random();
  while (n != 1)          // this loop terminates
     n = syracuse(n);
}
```

Hard, in theory …

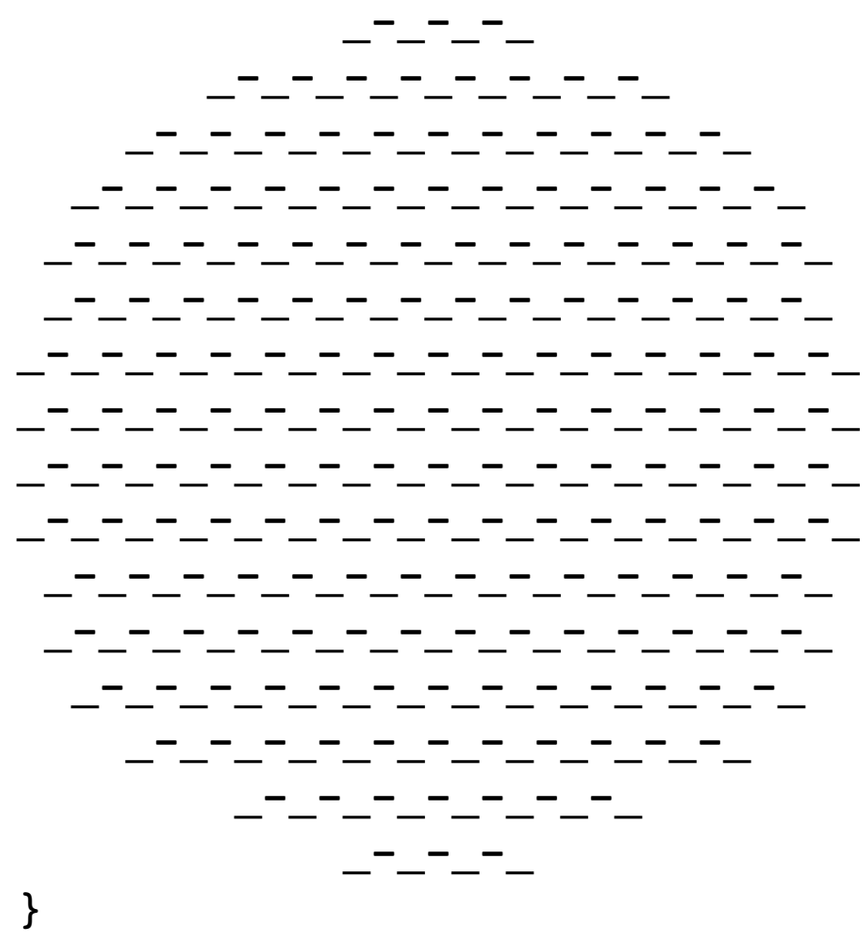
```
int syracuse(int n) {
  if (even(n))
    return n/2;
  return 3*n+1;
}

main() {
  int n = random();
  while (n != 1)          // this loop terminates
     n = syracuse(n);     // no one knows why
}
```

Hard, in practice …

Hard, in practice …

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.*-F/OO/
OO);}F_OO()
{
```



```
}
```

Still, there is hope!

- Software verification is undecidable ...

- Even decidable sub-problems are intractable ...

Still, there is hope!

- Software verification is undecidable ...
- But so is theorem proving!
- Even decidable sub-problems are intractable ...

Still, there is hope!

- Software verification is undecidable ...
- But so is theorem proving!
- Even decidable sub-problems are intractable ...
- But so is SAT solving!

Tool competitions

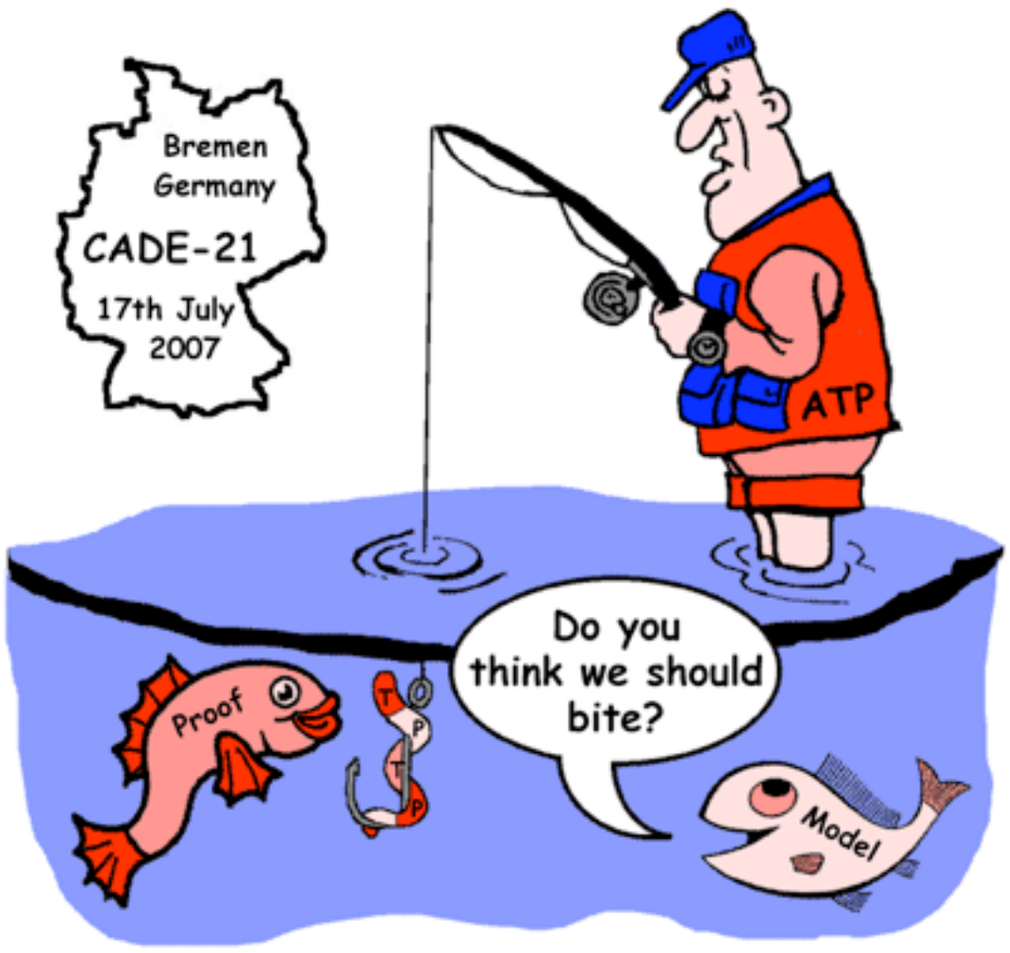
Tool competitions

Tool competitions

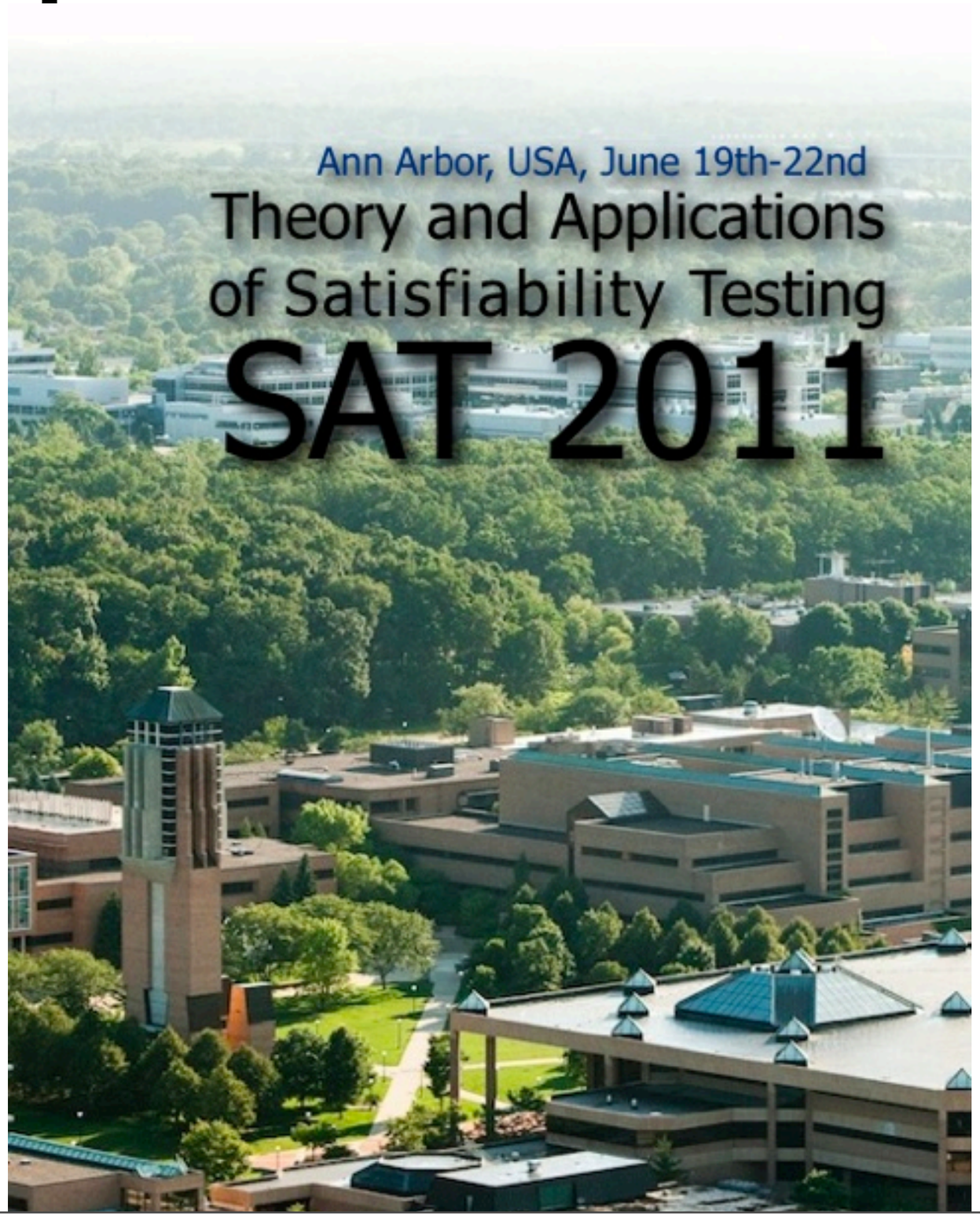
- ☑ Build communities
- ☑ Catalyze tool development
- ☑ Encourage cross-fertilization

Tool competitions

Tool competitions



CASC-21



Need to focus on ...

- Common input format
- Well-defined semantics
- Clearly stated decision problems

Need to focus on ...

- Common input format
- Well-defined semantics
- Clearly stated decision problems

what is safety ?

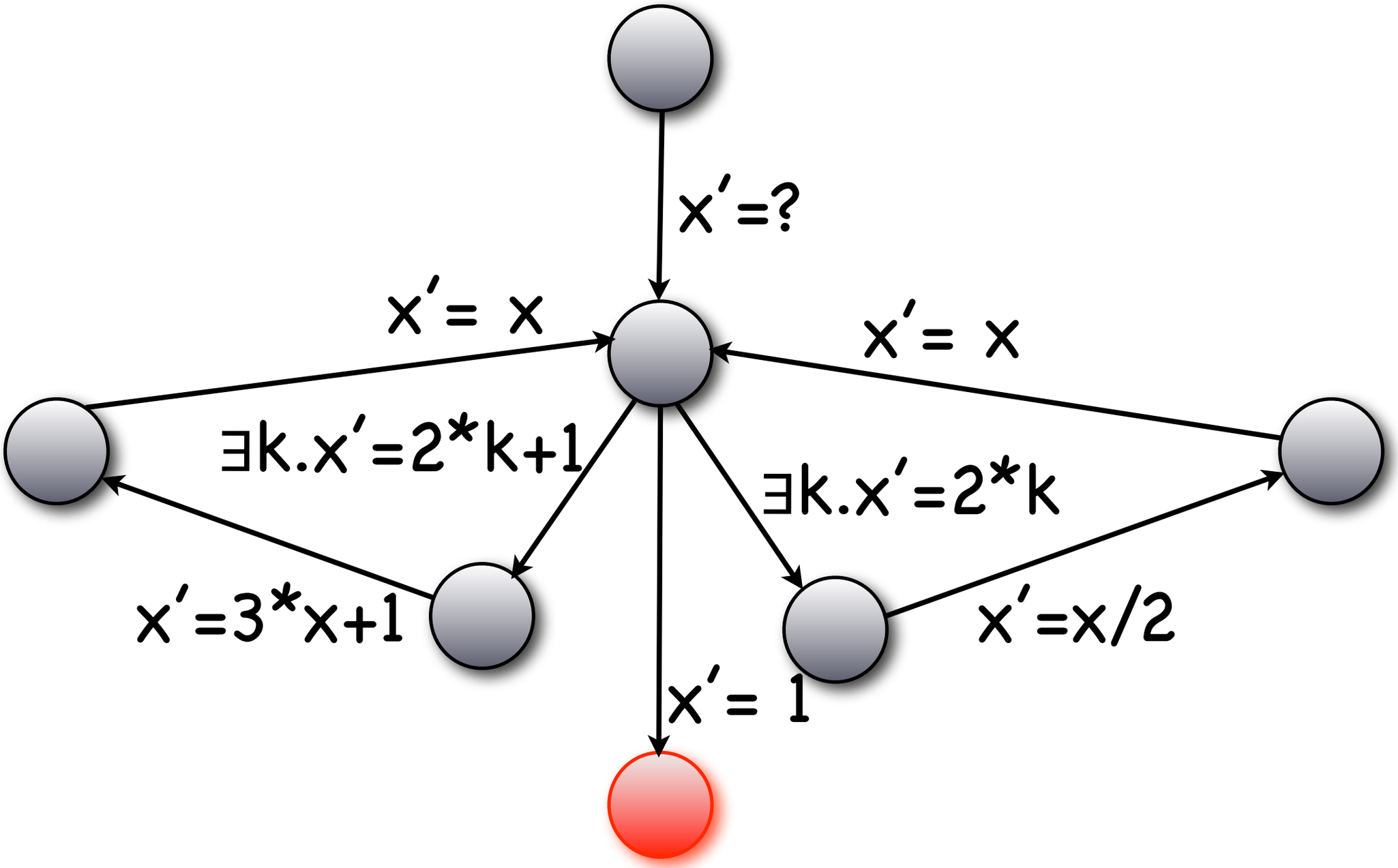
Need to focus on ...

- Common input format
- Well-defined semantics
- Clearly stated decision problems

what is safety ?

what is termination ?

Numerical Transition Systems



Numerical Transition Systems

- control states: initial, final, error
- typed variables: bool, nat, int, real
- transition rules: first-order logic

Numerical Transition Systems

- control states: initial, final, error
- typed variables: bool, nat, int, real
  - transition rules: first-order logic

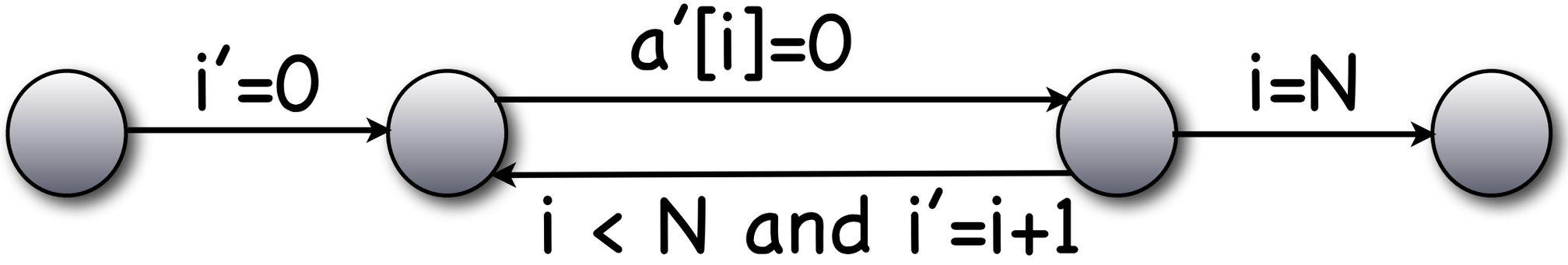
- safety: no error state is ever reached
- termination: a final state is eventually reached

Towards "real" programs

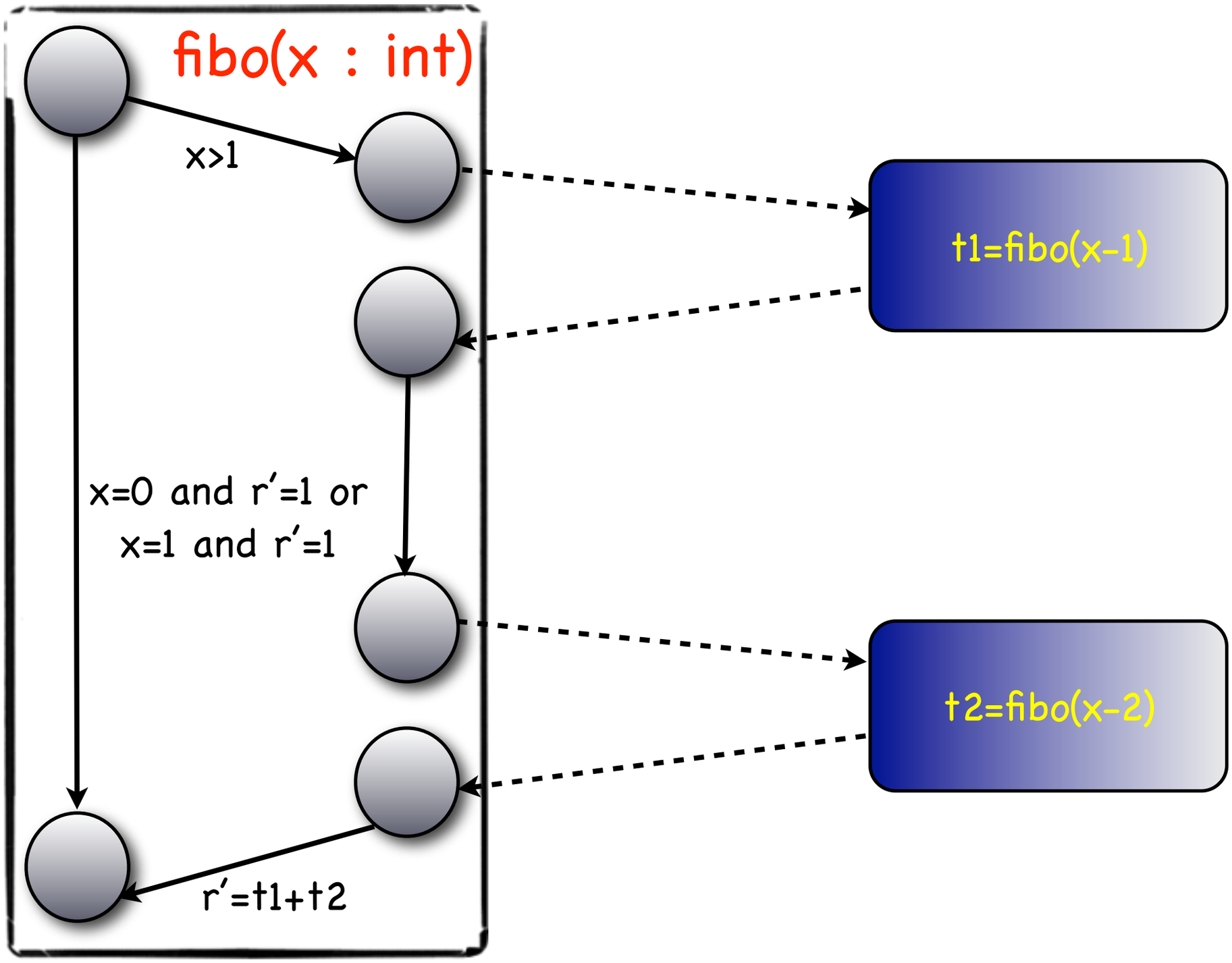
- array variables (arrays = functions)
- hierarchy and recursion
- parallelism (shared memory)

Array NTS

par N : int;
a[N], i : int;

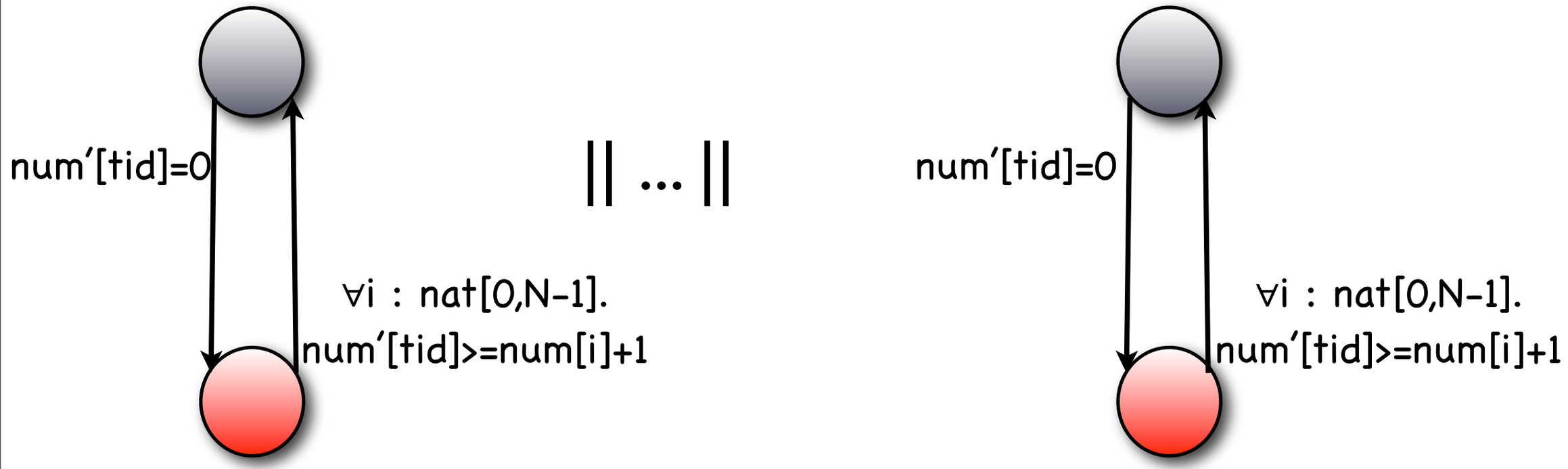


Hierarchic/Recursive NTS



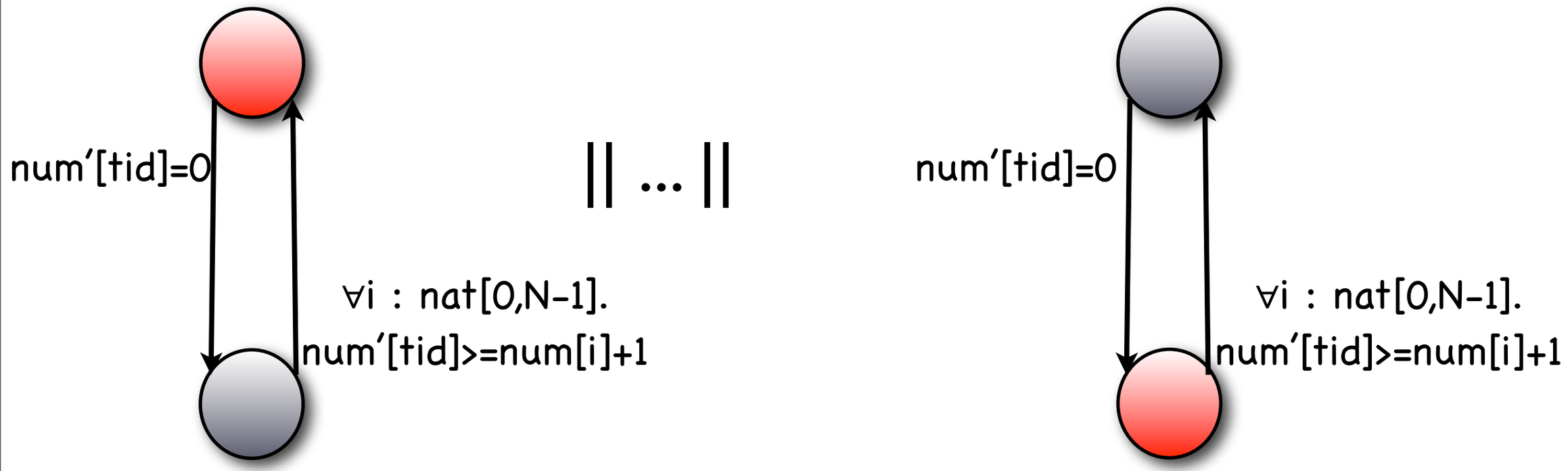
Parallel NTS

par N : int;
num[N] : int;



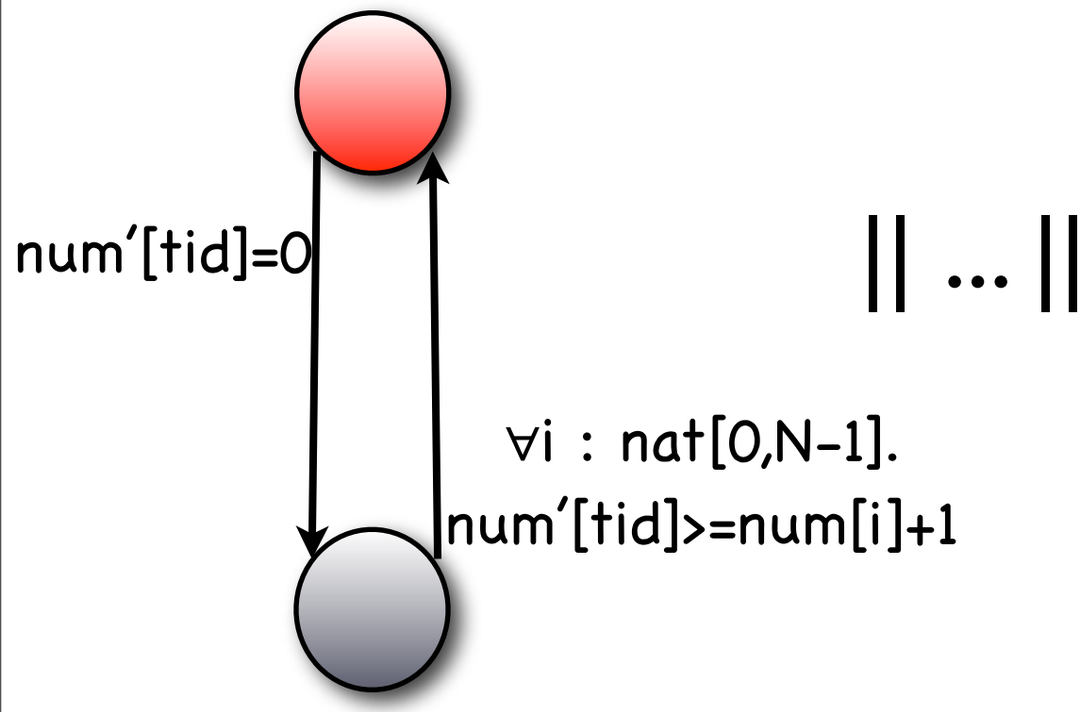
Parallel NTS

par N : int;
num[N] : int;

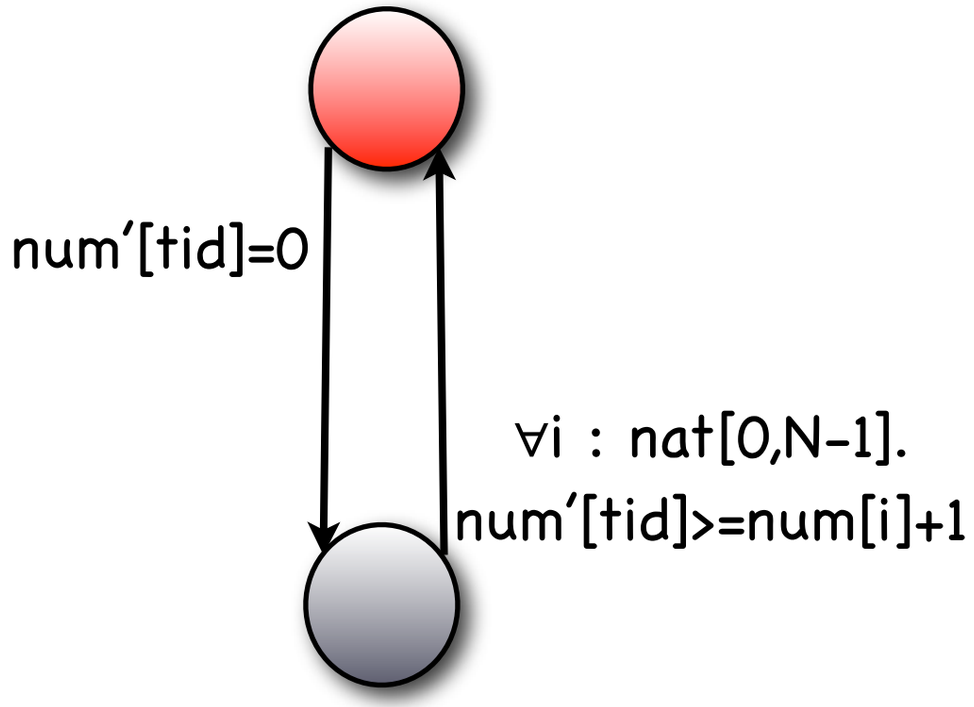


Parallel NTS

par N : int;
num[N] : int;

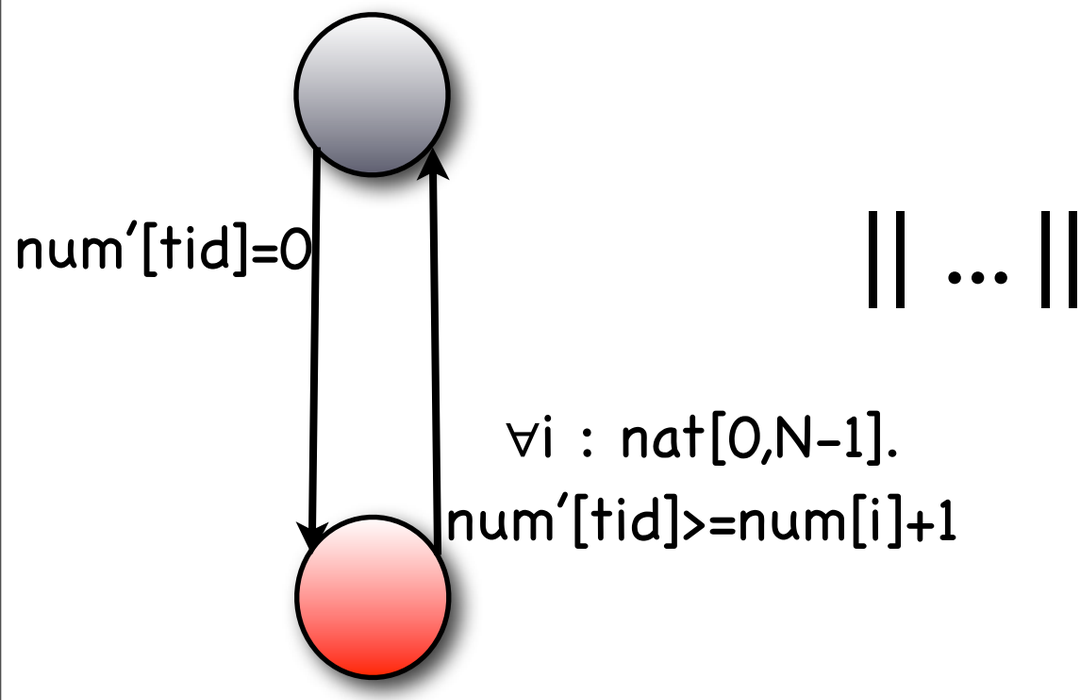


|| ... ||

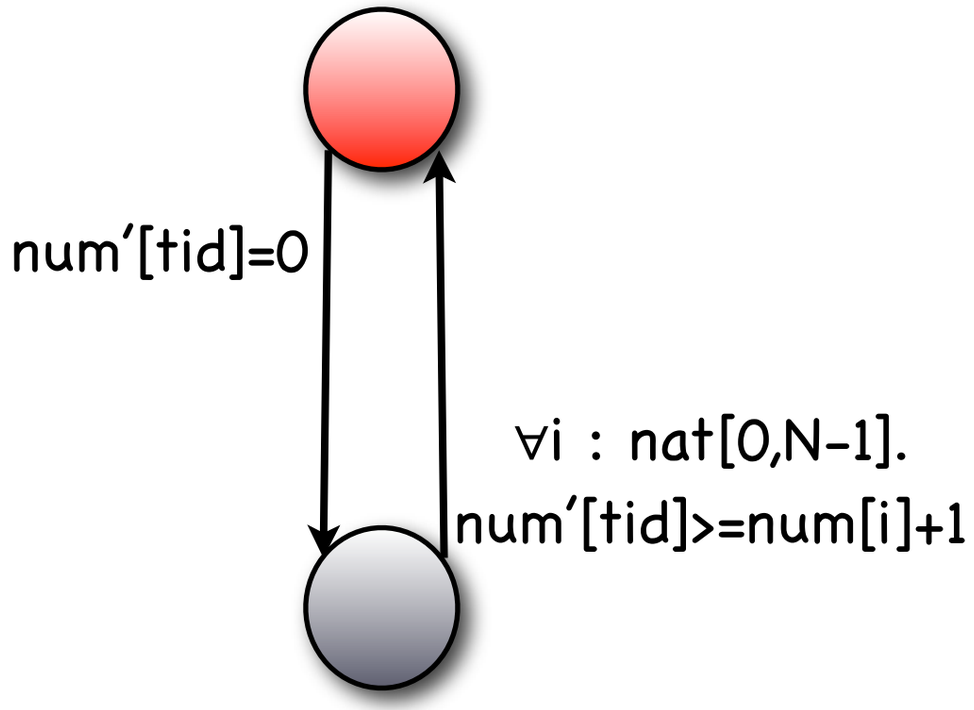


Parallel NTS

par N : int;
num[N] : int;

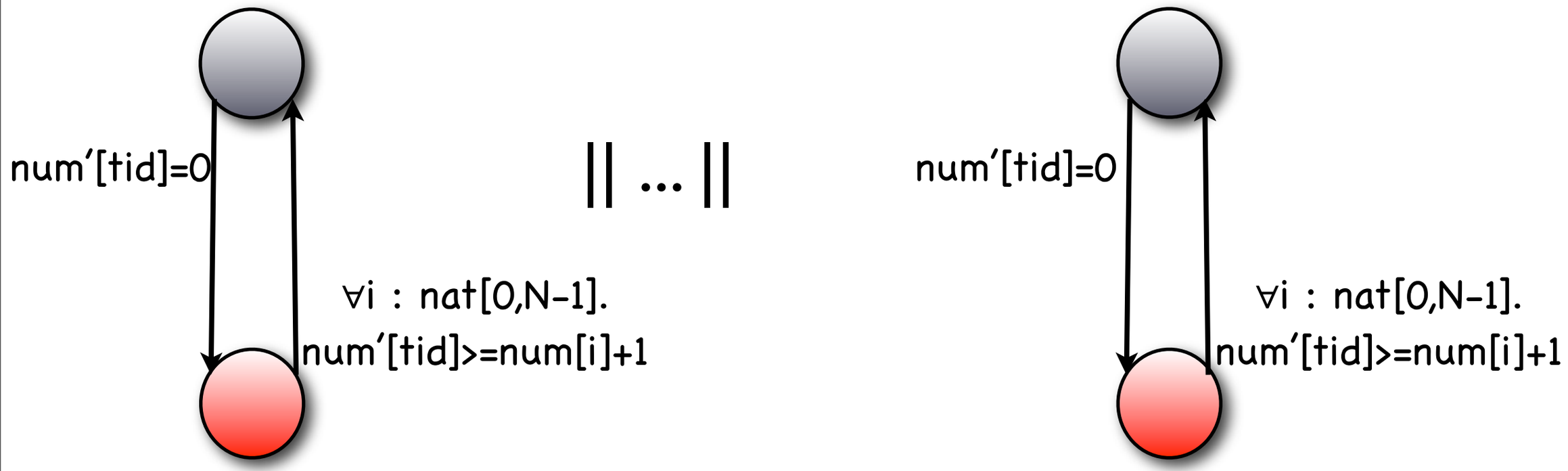


|| ... ||



Parallel NTS

par N : int;
num[N] : int;



Building a library of benchmarks

Building a library of benchmarks

floppy.c

apache.c

firewire.c

⋮

Building a library of benchmarks

floppy.c

apache.c

firewire.c

⋮

peterson

bakery

dekker

⋮

Building a library of benchmarks

floppy.c

apache.c

firewire.c

⋮

peterson

bakery

dekker

⋮

bench1.nts — affine, recursive, array

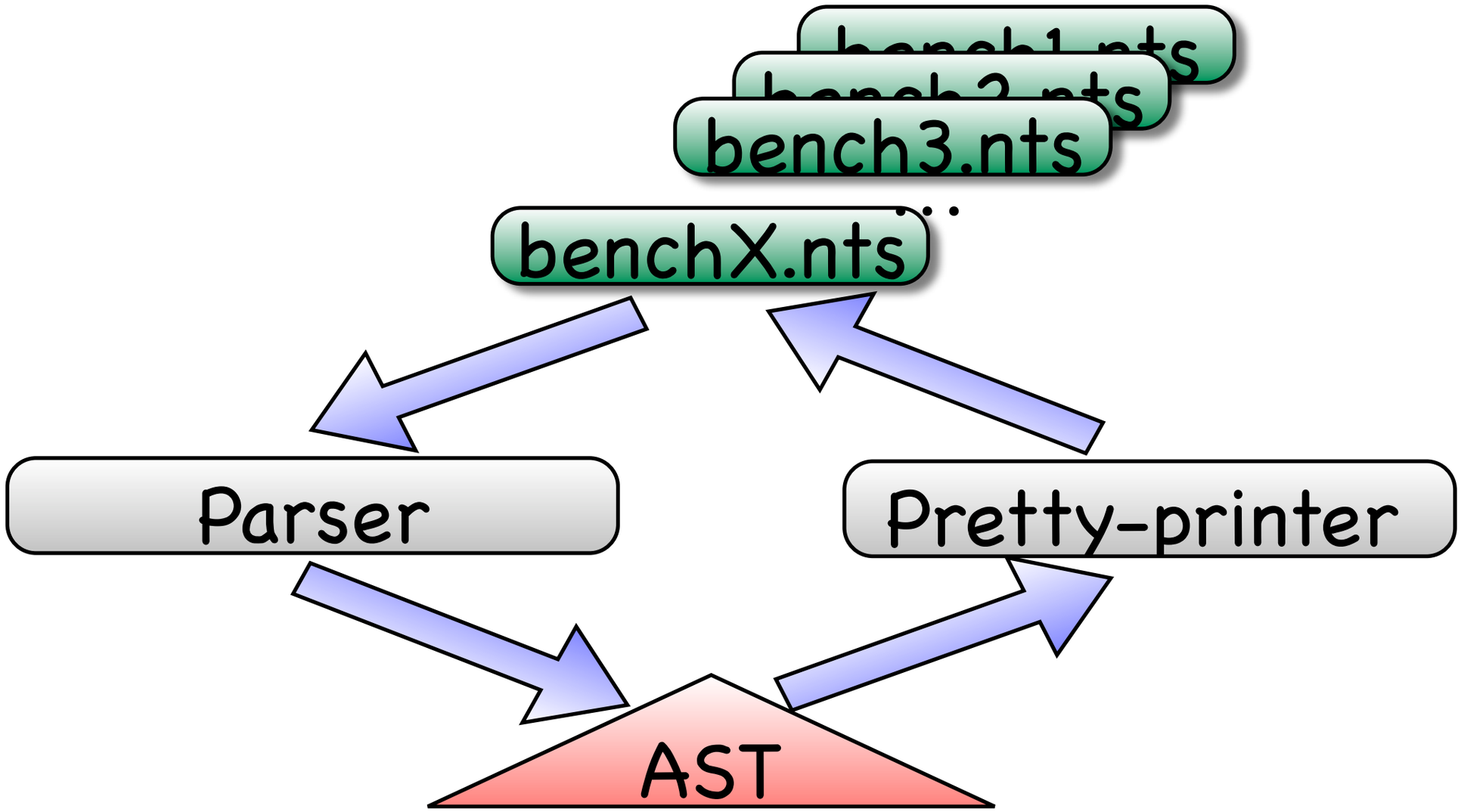
bench2.nts — octagon, parallel, array

bench3.nts — flat, octagon, int

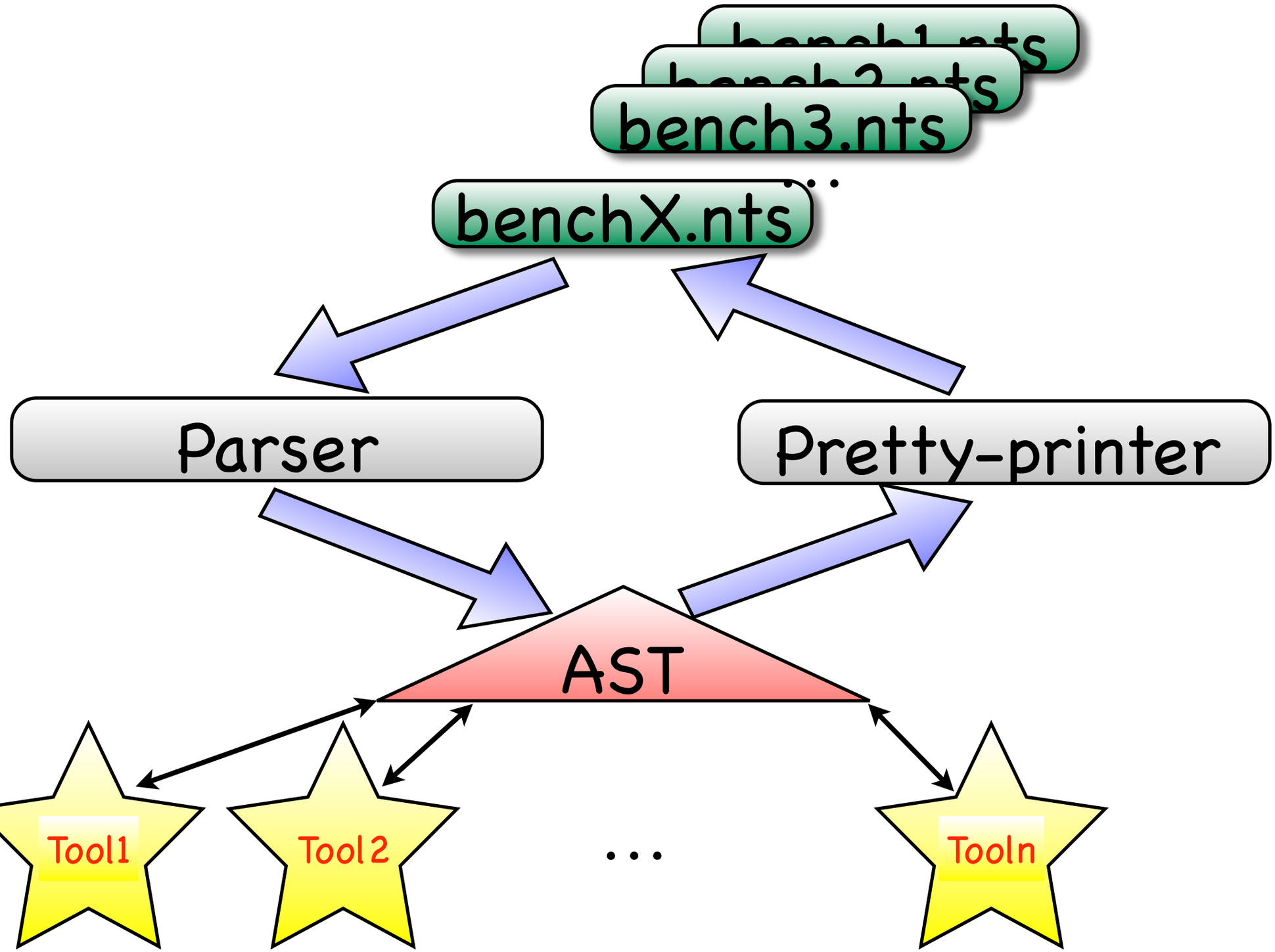
...

benchX.nts — real, affine, hierarchy

Why NTS-LIB ?

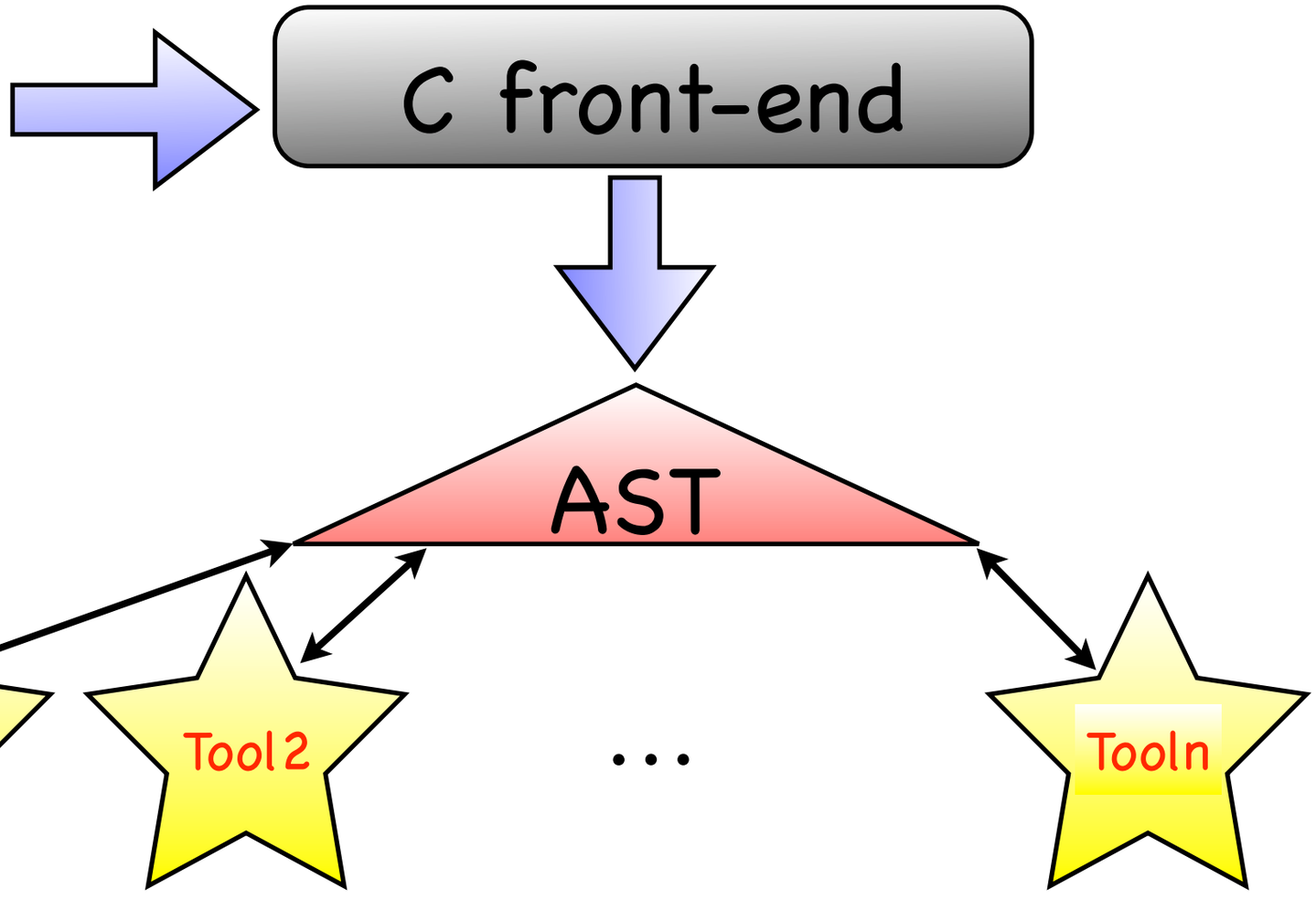


Why NTS-LIB ?


bench3.nts
benchX.nts
Parser
Pretty-printer
AST
Tool1
Tool2
...
Tooln

Cooperation vs. competition

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main()
{F_OO();printf("%1.3f\n",4.*-F/OO/
OO);}F_OO()
{
          _-_-_-_
       _-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_
   _-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_
      _-_-_-_-_-_-_-_-_
        _-_-_-_-_-_-_
          _-_-_-_
}
```



Concrete actions

- ☑ Release NTS-LIB (Java/OCaml)
- ☑ C front-end based on CIL/Frama-C
- ☑ Bridge existing tools (ARMC, Fast, Flata, INTERPROC, ASPIC, etc.)
- ☑ Gather benchmarks and tag them
- ☑ Create sections:
  - affine, recursive, parallel, ...
  - safety, termination ...

Start the competition!