Extending the RealySt tool to support linear hybrid automata

Final Talk

Dilara Arslan
Supervision: Prof. Dr. Dr.h.c. Erika Ábrahám
Apl. Prof. Thomas Noll

LuFG Theory of Hybrid Systems

SS 2026

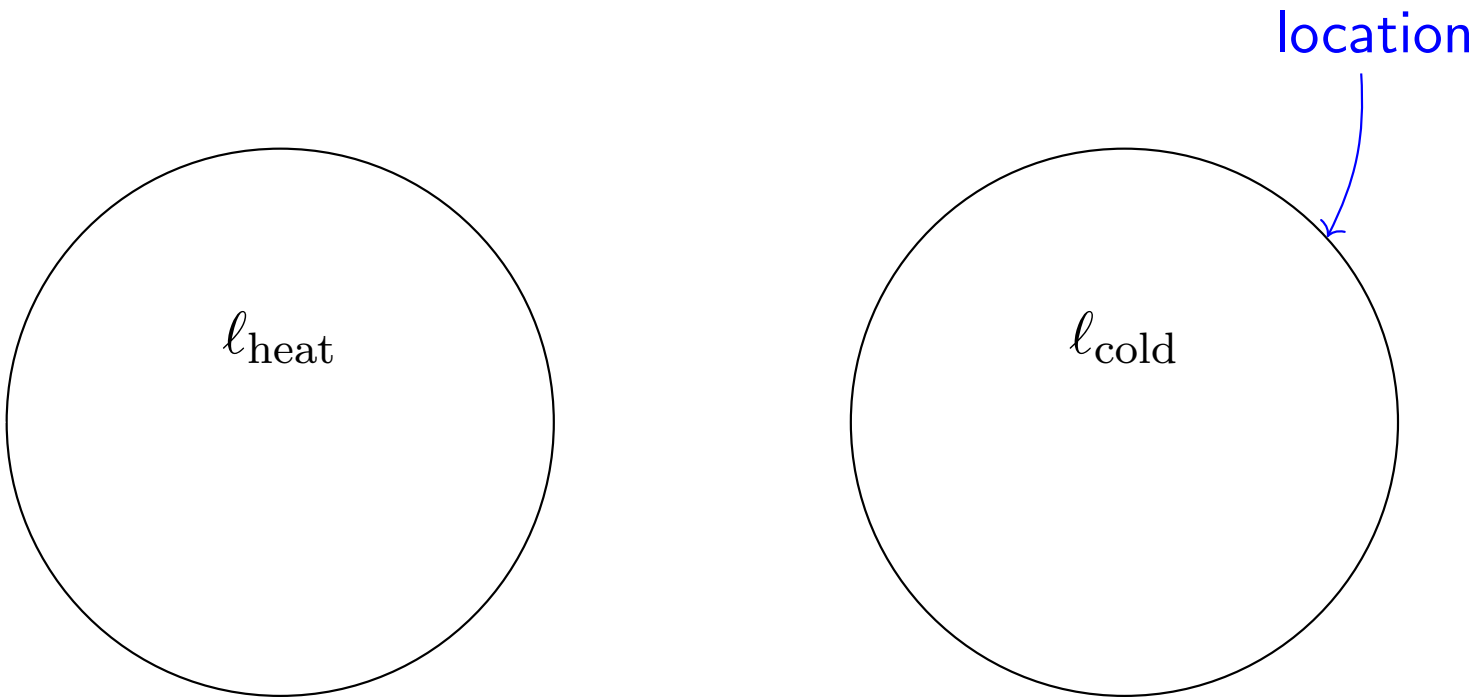
Motivation

- Hybrid automata describe physical phenomena
- Stochastic hybrid automata add probabilistic event
- Several tools which can analyze automata
- JANI was introduced since interoperability was limited
- REALYST only supported rectangular hybrid automata
- Linear hybrid automata are more expressive

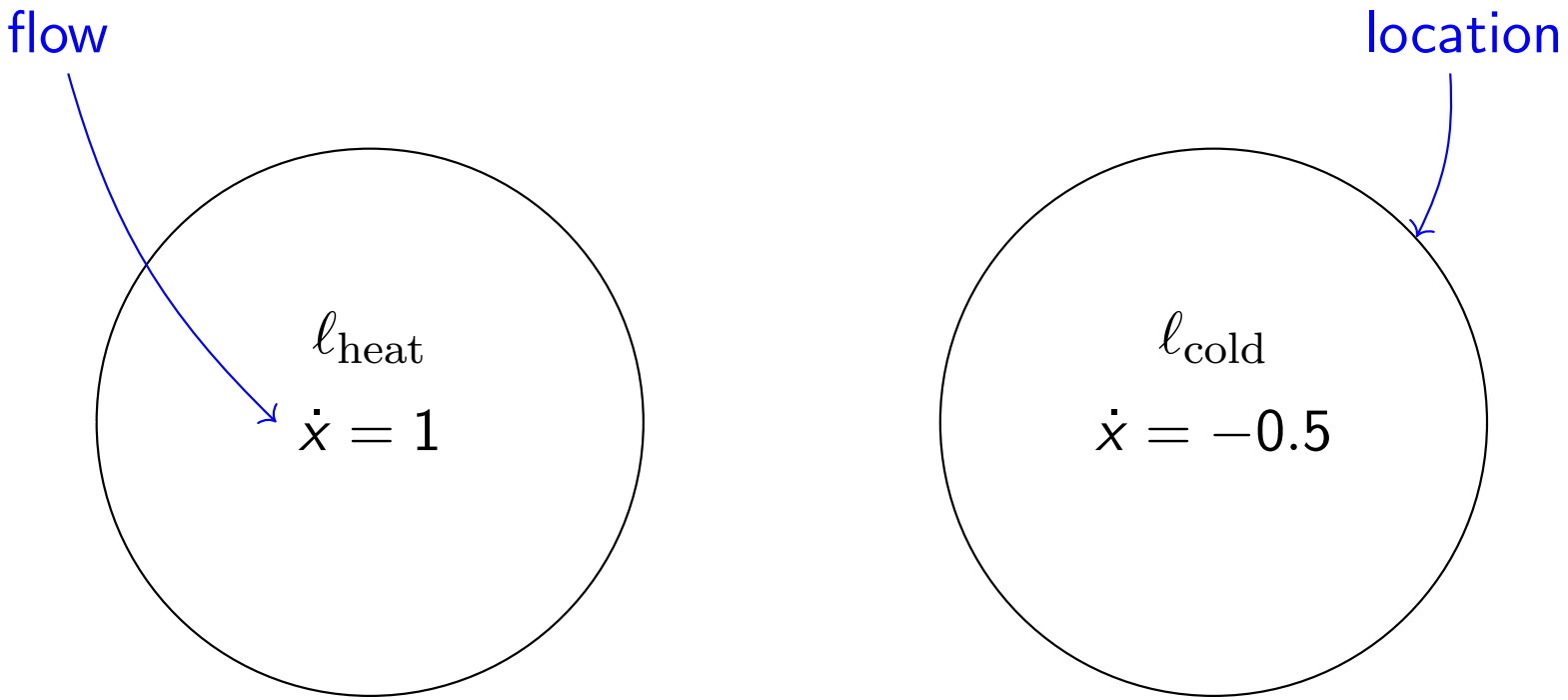
Outline

1. Hybrid Automata
2. Reachability Analysis
3. Tools
4. JANI
5. Implementation
6. Results

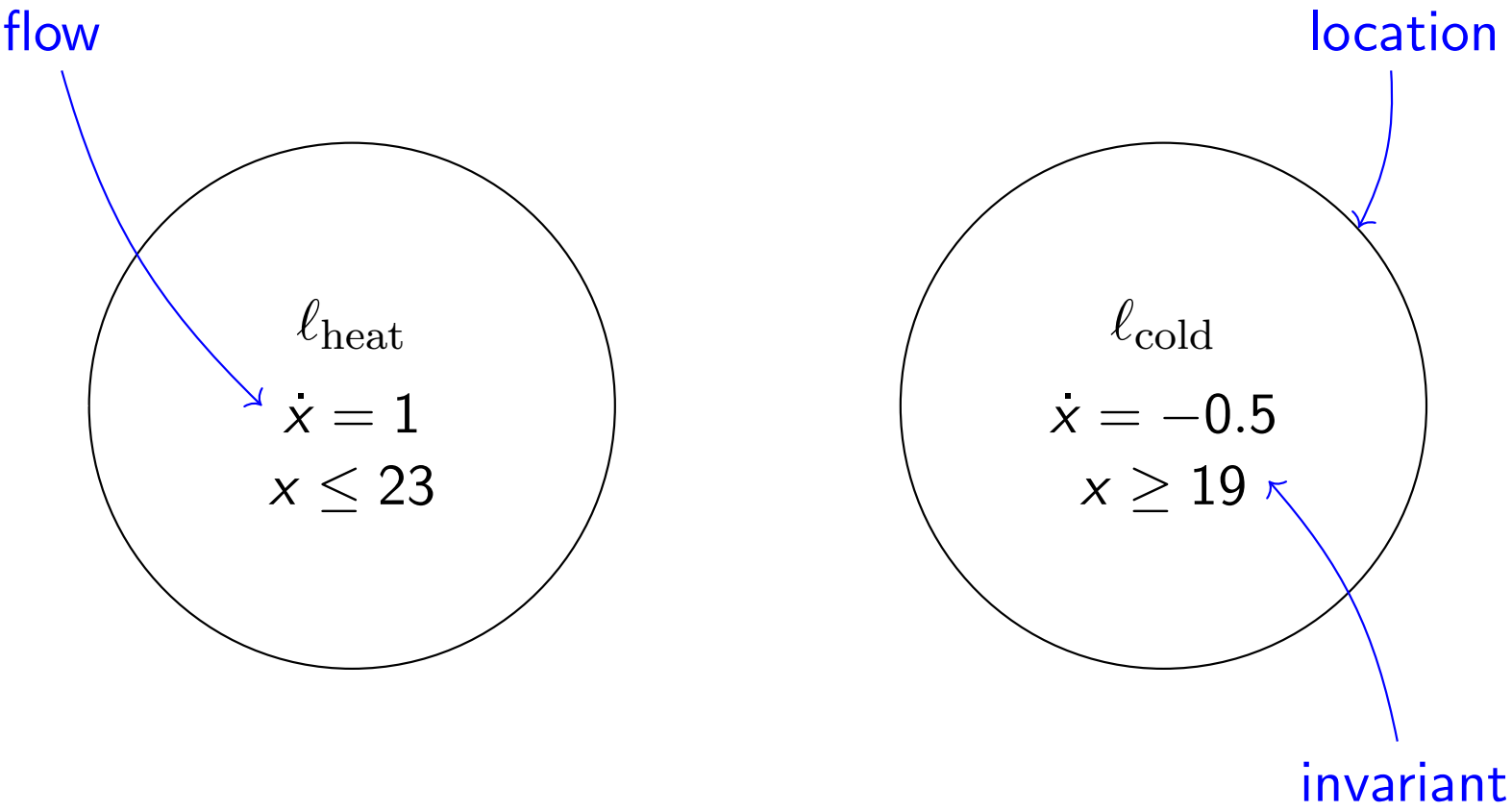
Example Hybrid Automaton



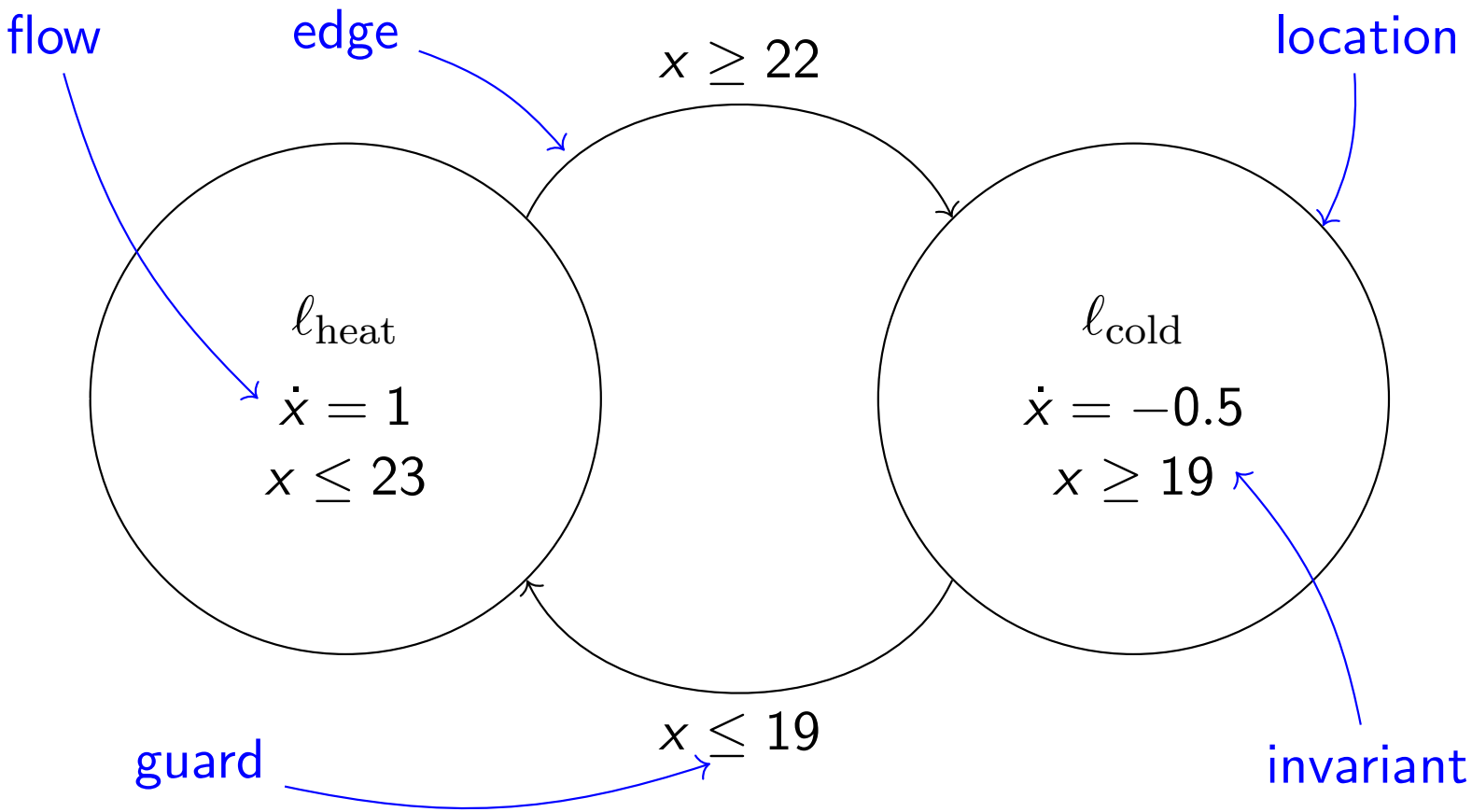
Example Hybrid Automaton



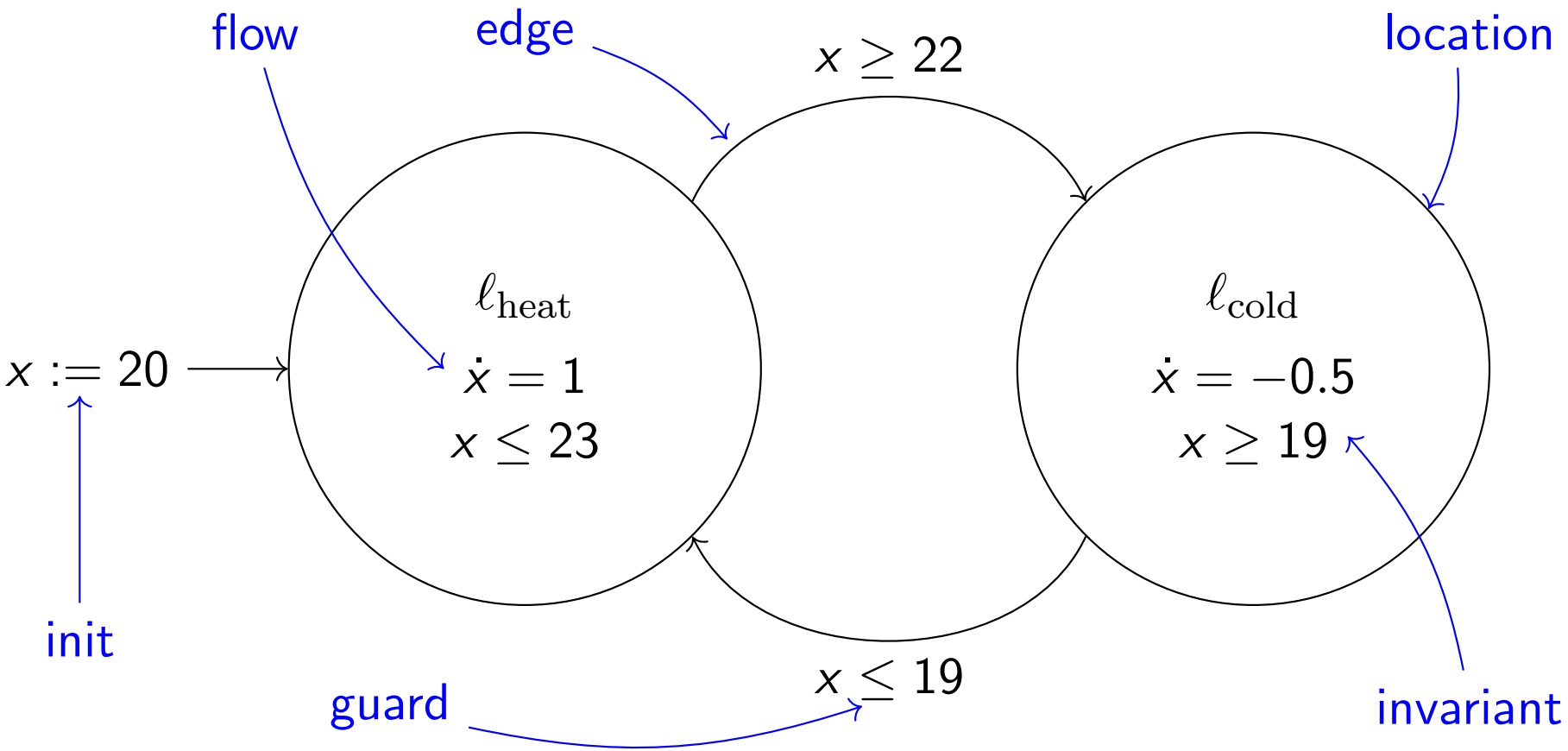
Example Hybrid Automaton



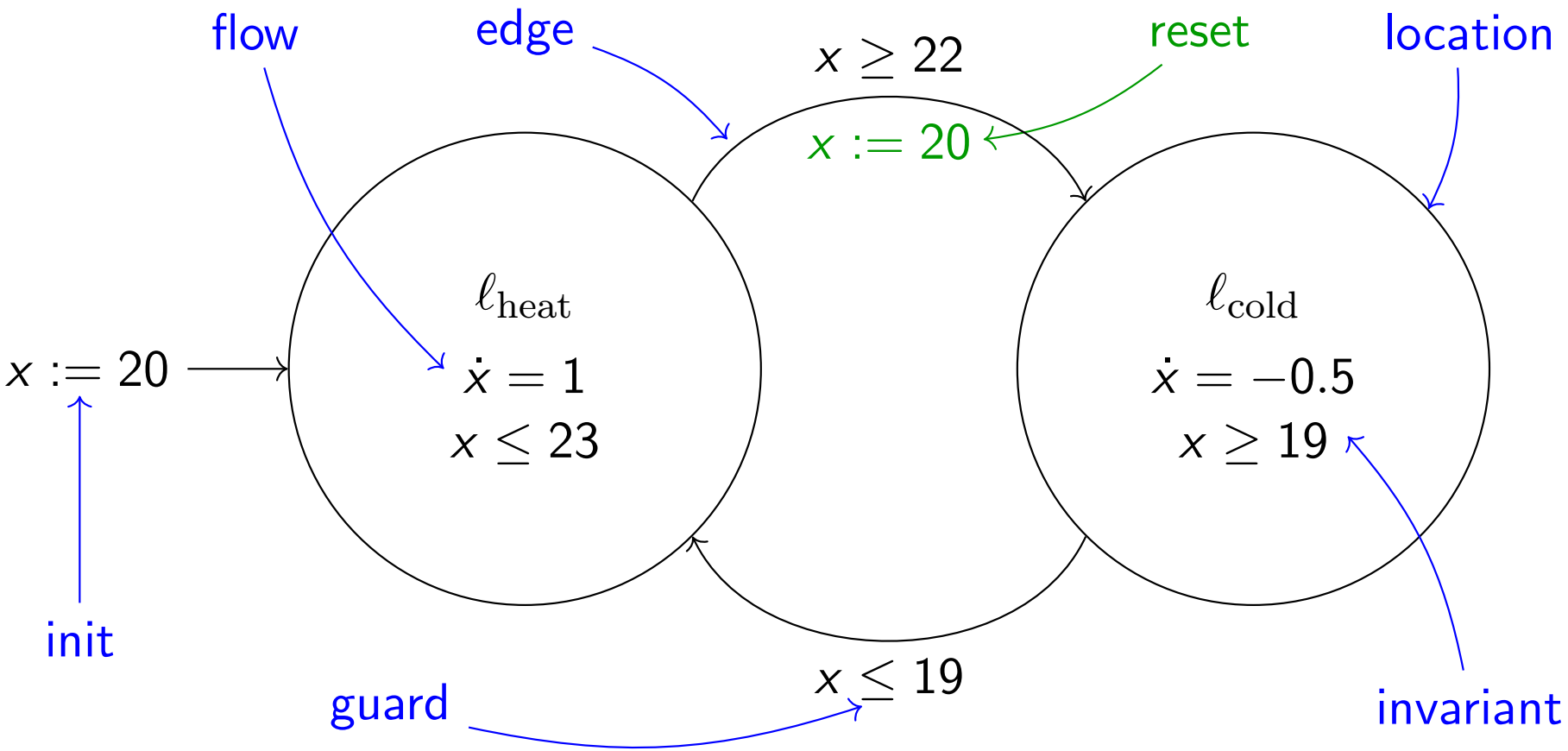
Example Hybrid Automaton



Example Hybrid Automaton



Example Hybrid Automaton



Subclasses of Hybrid Automata

Rectangular Hybrid Automata:

- Flows, invariants, guards and resets are defined by rectangular sets
- $[a, b]$ represents the rectangular set for a variable x with $a \leq x \leq b$

Linear Hybrid Automata:

- Invariants, guards and resets are defined either by a linear inequality or rectangular set
- Type I: flows are defined as rectangular sets
- Type II: flows are defined as linear ordinary differential equations $\dot{x} = A * x + b$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^n$

Subclasses of Hybrid Automata

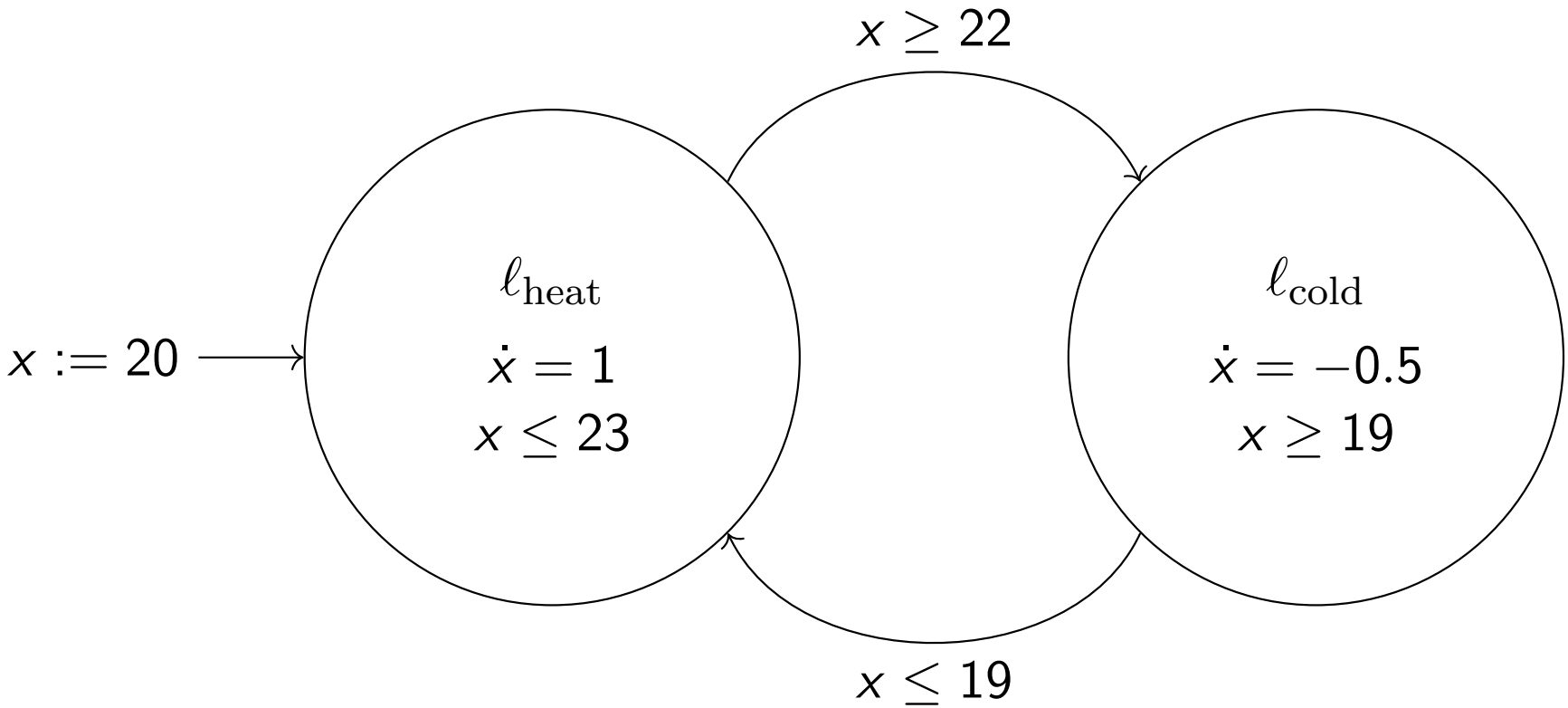
Rectangular Hybrid Automata:

- Flows, invariants, guards and resets are defined by rectangular sets
- $[a, b]$ represents the rectangular set for a variable x with $a \leq x \leq b$

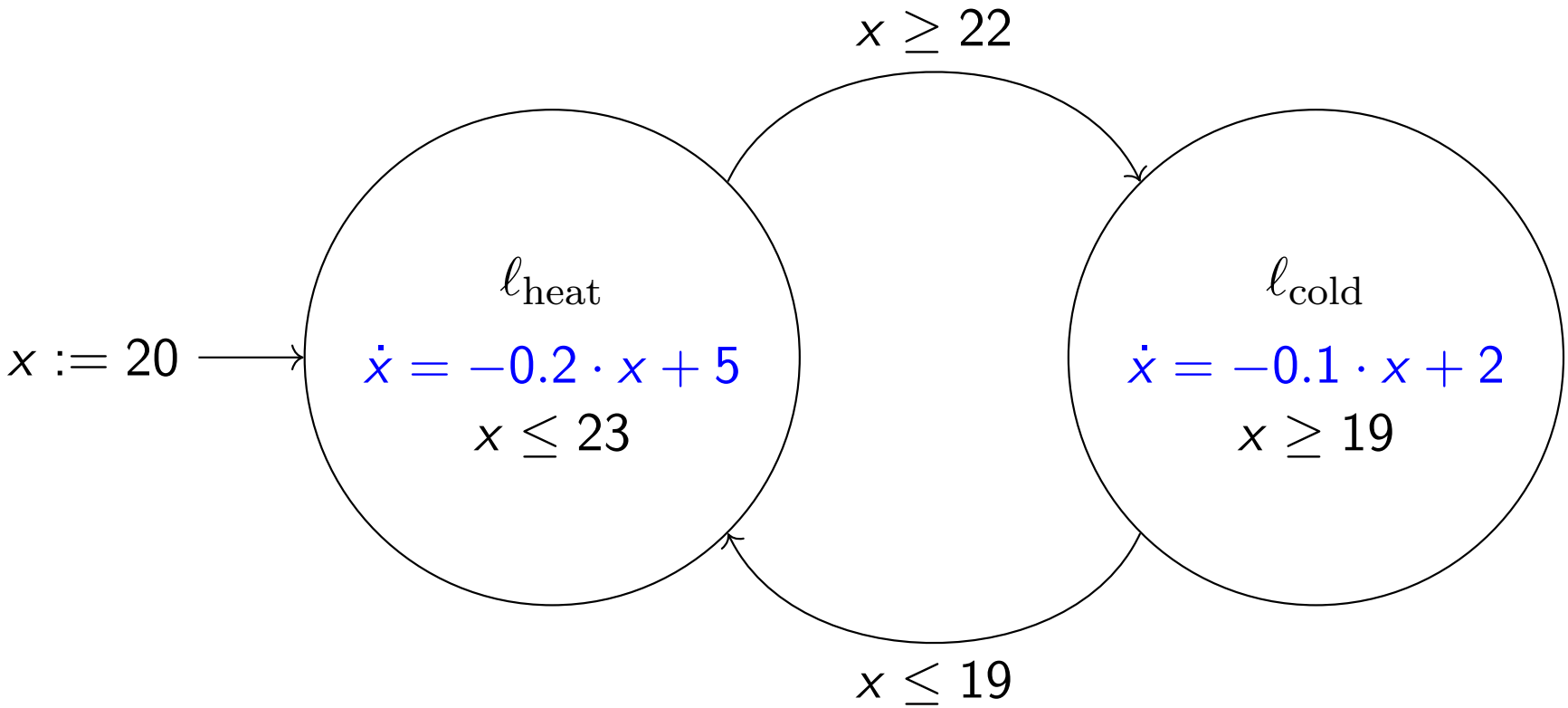
Linear Hybrid Automata:

- Invariants, guards and resets are defined either by a linear inequality or rectangular set
- Type I: flows are defined as rectangular sets
- Type II: flows are defined as linear ordinary differential equations $\dot{x} = A * x + b$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^n$

Example Linear Hybrid Automaton II



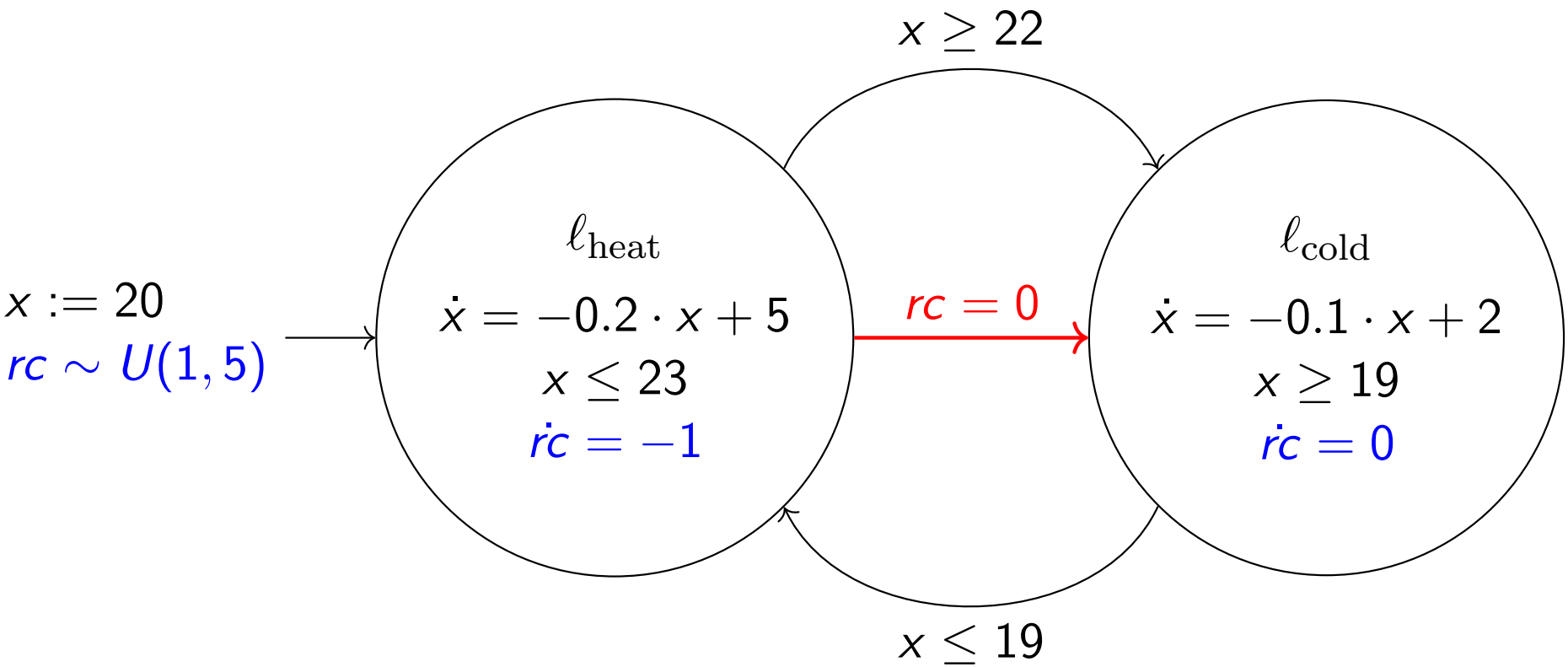
Example Linear Hybrid Automaton II



Linear Hybrid Automata II with Random Clocks

- Random clocks model nondeterministic choices
- Stochastic transitions fire when clocks expire
- Enables modeling uncertainty in systems

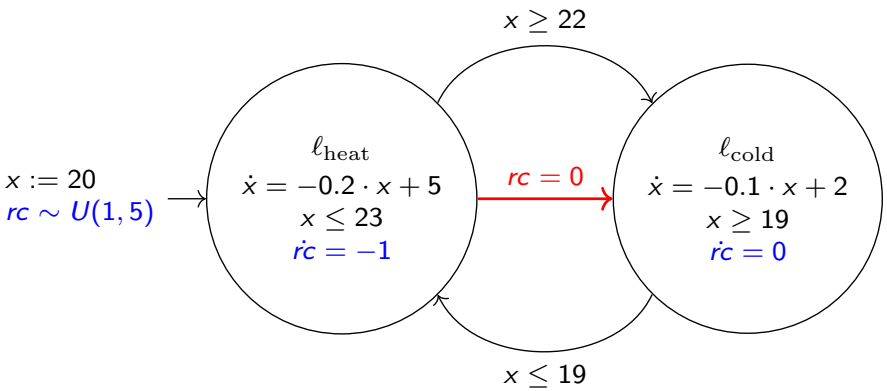
Example Linear Hybrid Automaton II with Random Clocks



Reachability Analysis

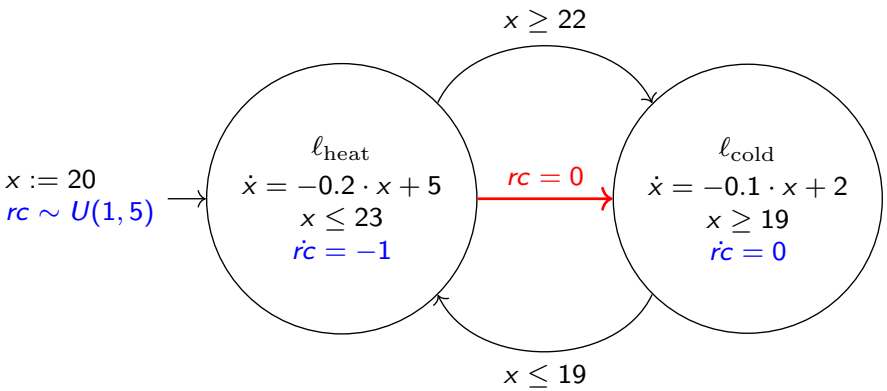
- Calculation how reachable a state from the initial location is
- Performed by iteratively computing the set of reachable states
- Problems: infeasible computation, Zeno execution
- Solutions: time-bound, limited jump depth

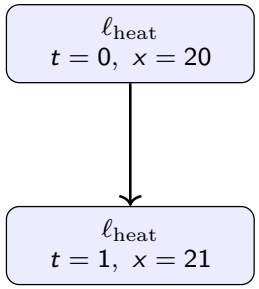
Reachability Tree for the Heating Example



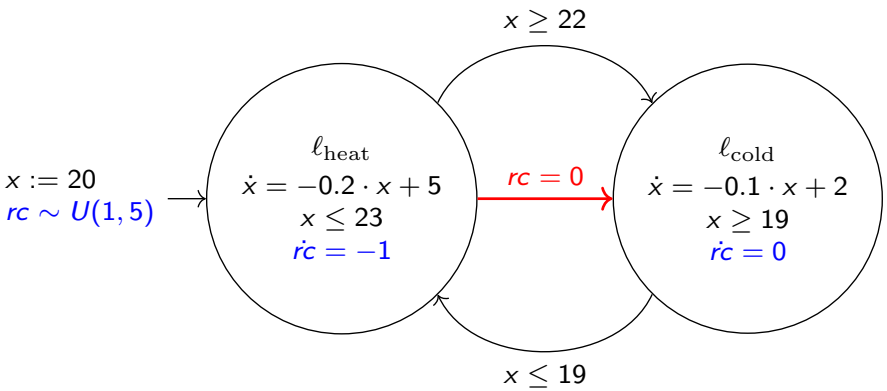
ℓ_{heat}
$t = 0,\ x = 20$

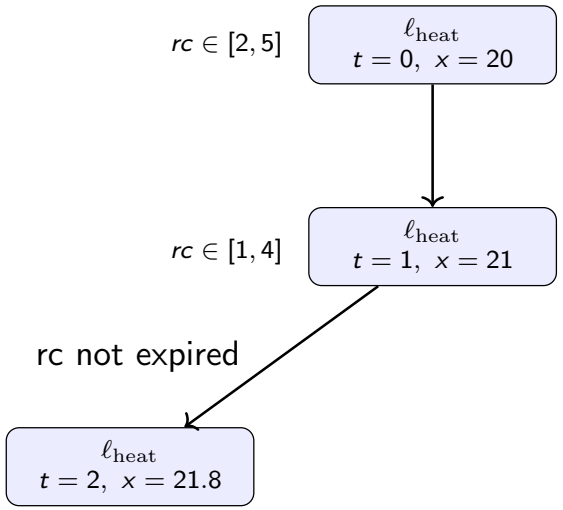
Reachability Tree for the Heating Example



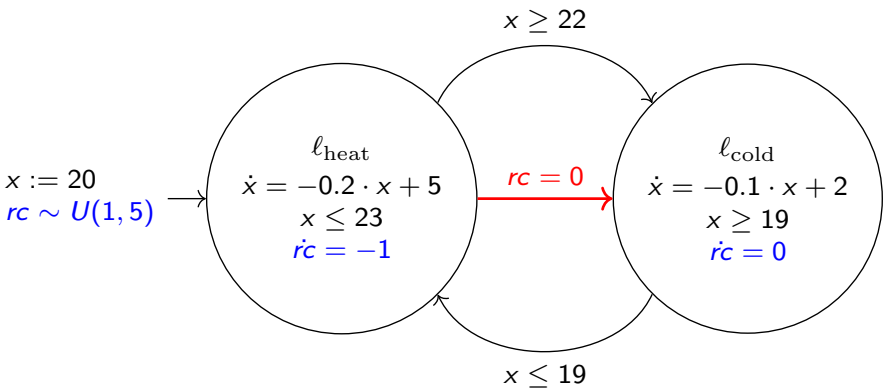


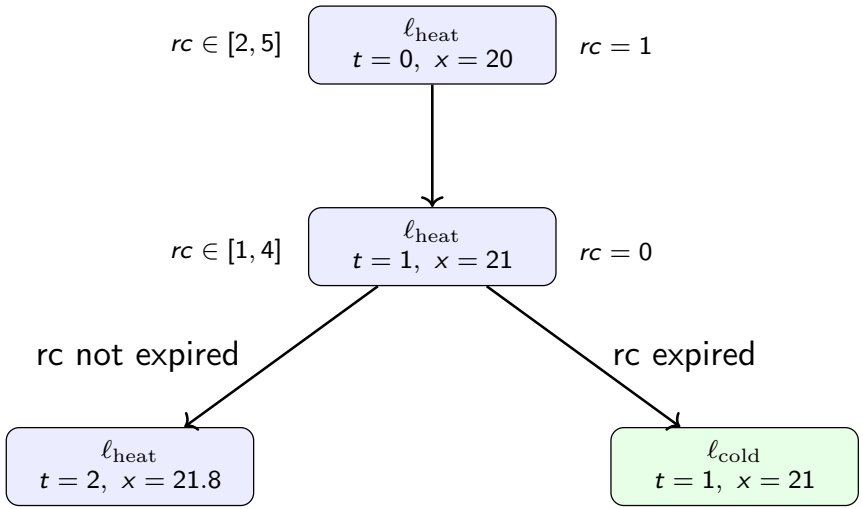
Reachability Tree for the Heating Example



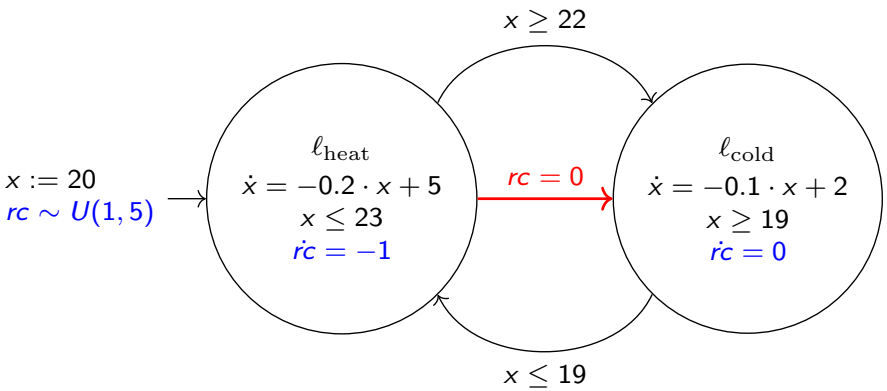


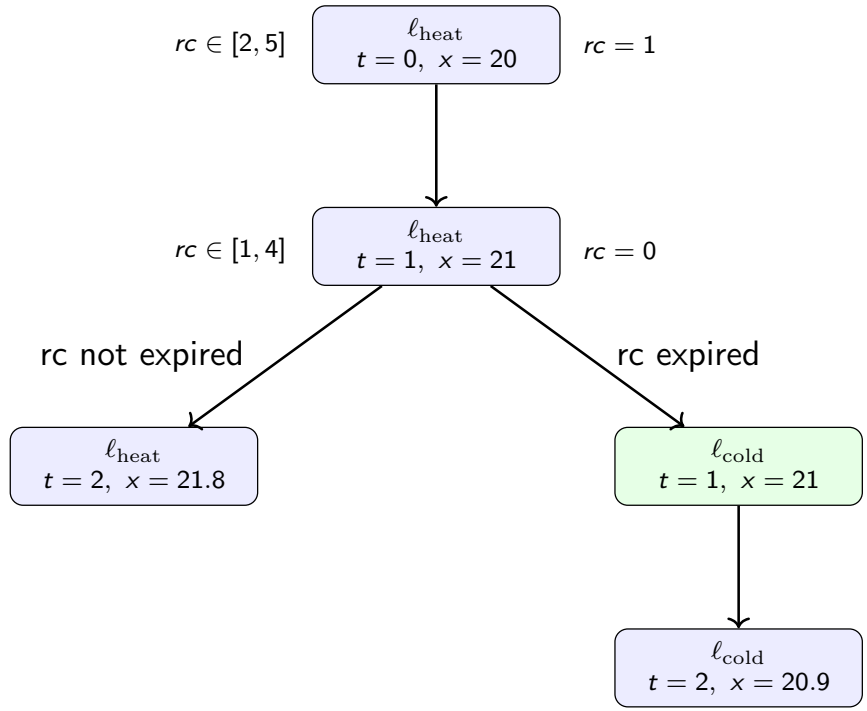
Reachability Tree for the Heating Example





Reachability Tree for the Heating Example





HyPro

- Provides implementations of state set representations for the reachability analysis
- Provides different geometric abstractions for polytopes
- Offers operations required for reachability analysis of linear hybrid automata

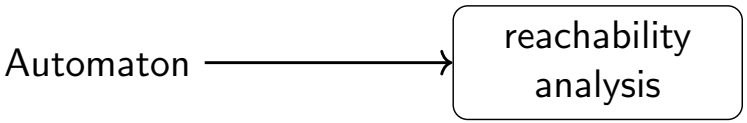
RealySt

- Open-source tool written in C++ for probabilistic reachability analysis
- Computes maximum reachability probabilities for singular automata with random clocks
- Uses HyPro for geometric operations and flowpipe construction

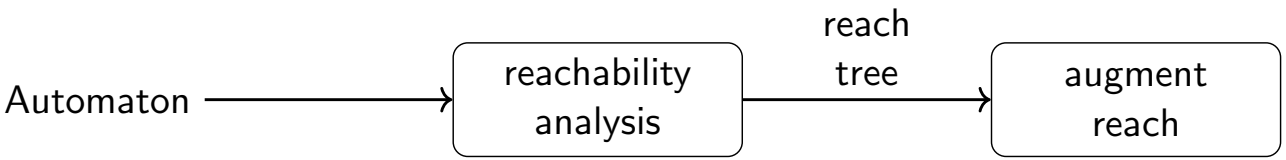
RealySt: Flow

Automaton

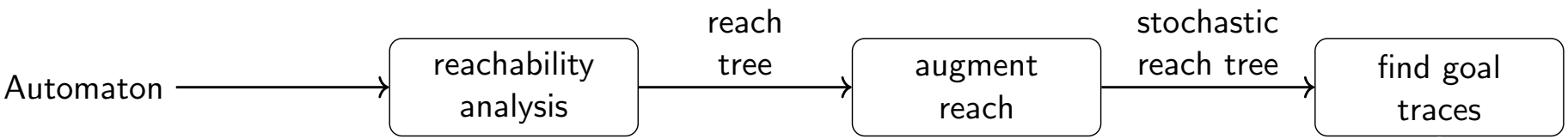
RealySt: Flow



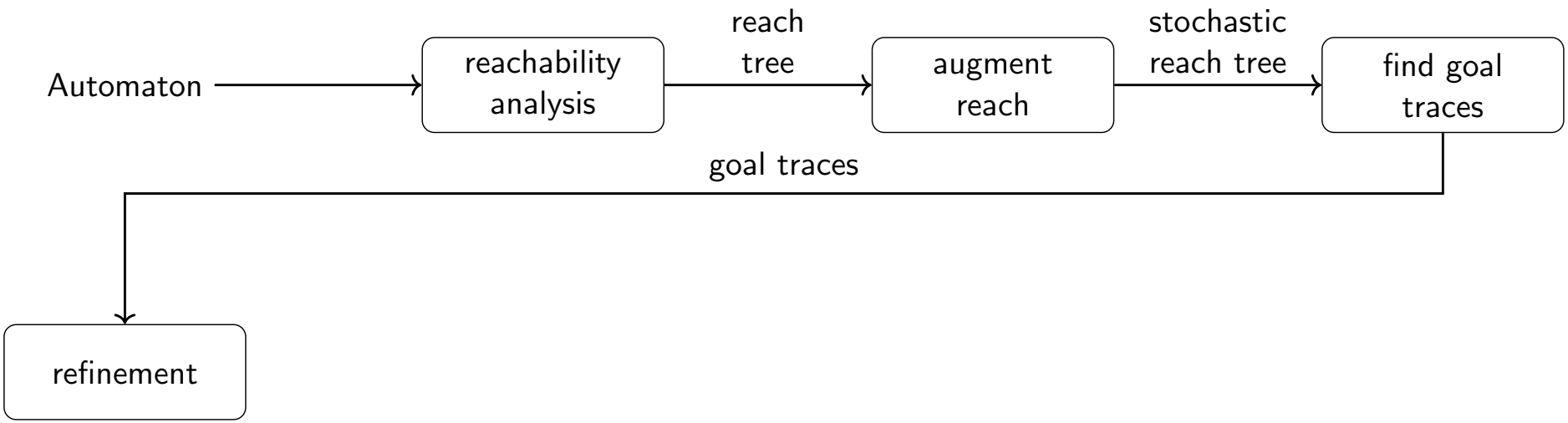
RealySt: Flow



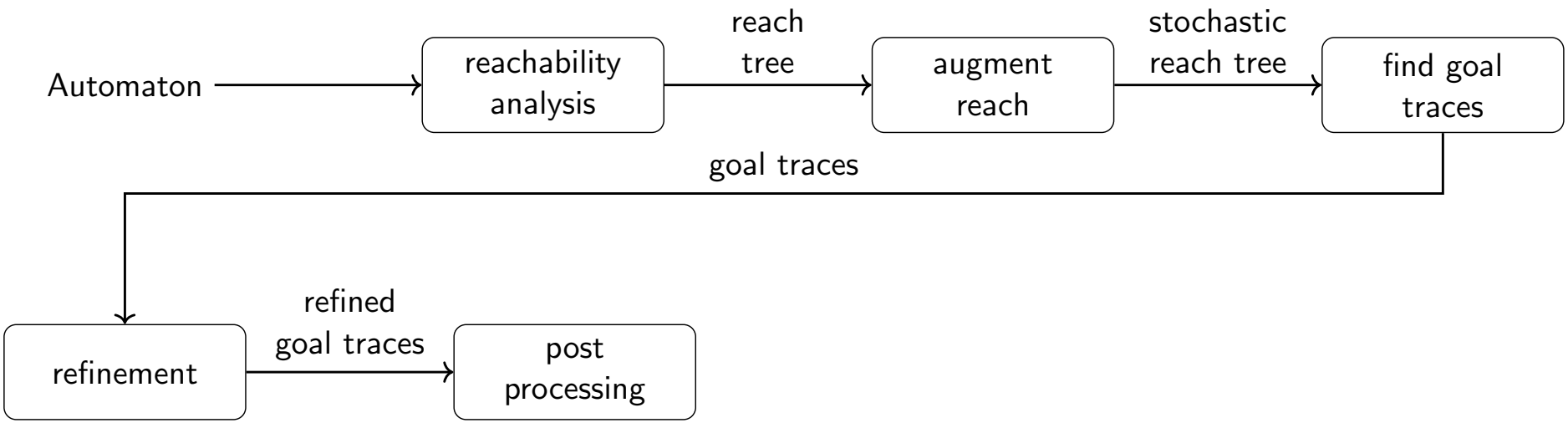
RealySt: Flow



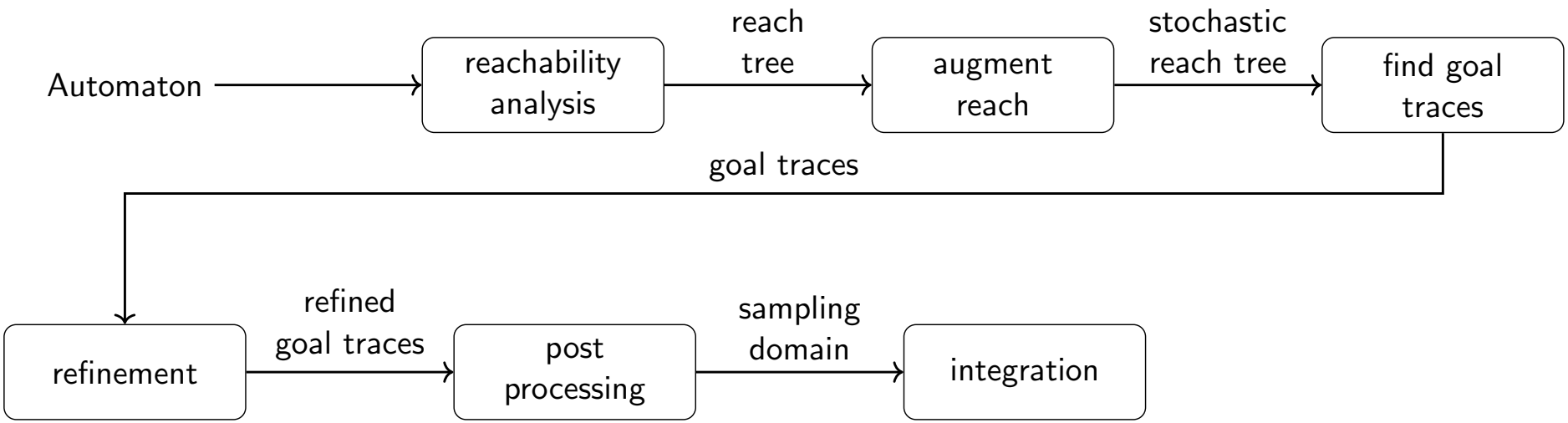
RealySt: Flow



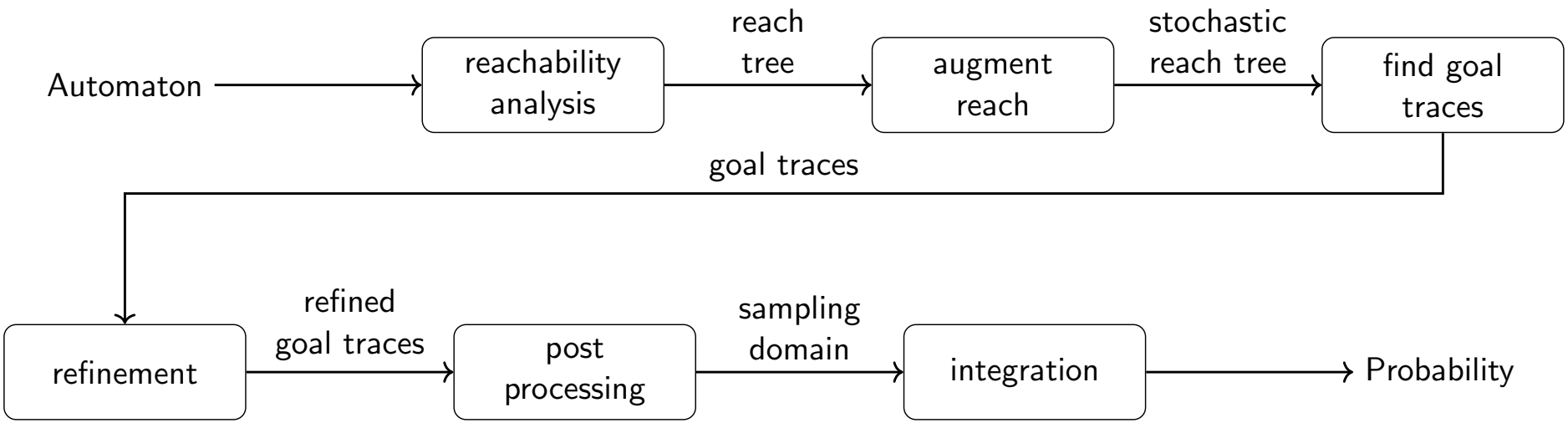
RealySt: Flow



RealySt: Flow



RealySt: Flow



RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- **`--LTIAnalysis`**
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

RealySt: Command Call

- `-t [timebound]`
- `-d [jumpdepth]`
- `-b [BENCHMARK]`
- `-m [CASE]`
- `--from-jani [FILEPATH]`
- `-l [info/trace/debug]`
- `--plotDimensions [index1] [index2]`
- `--LTIAnalysis`
- `--createJani [FILENAME]`

JANI

- Standardized exchange format
- JSON data format to encode models and messages
- Enables interoperability between tools
- Design also allows for future extensions

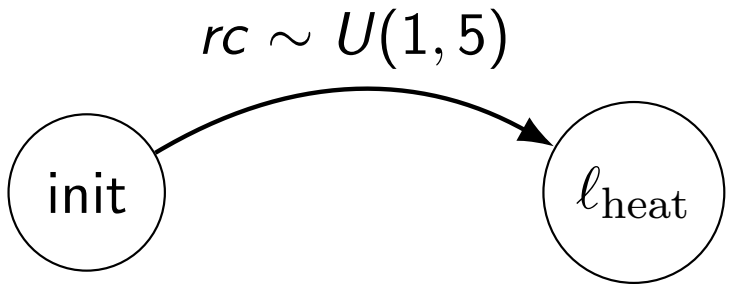
JANI: Model Header and Variables

```
{
 "jani-version": 1,
 "name": "heating_rc_example",
 "type": "sha",

 "variables": [
 { "name": "x", "type": "continuous",
 "initial-value": 20.0 },
 { "name": "rc", "type": "continuous" }
 ]
}
```

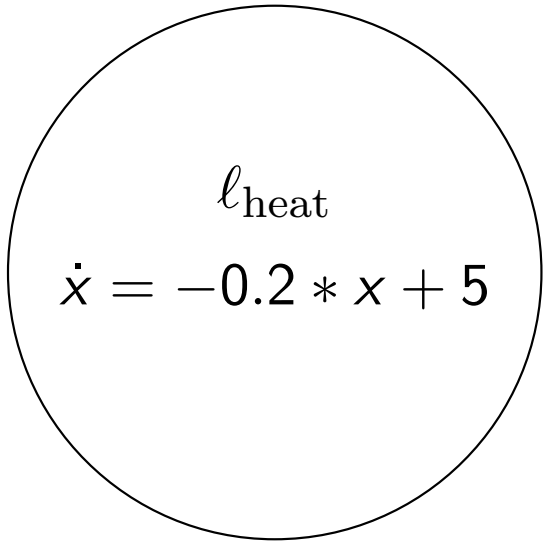
JANI: Stochastic Initialization

```
"location": "init_stochastic_variables",
"destinations": [
 {
  "assignments": [
   {
    "ref": "rc",
    "value": {
     "args": [1.0, 5.0],
     "distribution": "Uniform"
    }
   }
  ],
  "location": "l_heat"
 }
]
```



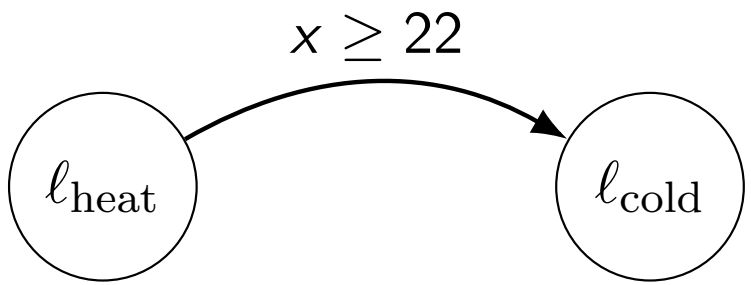
JANI: Location with Linear Flow

```
"name": "l_heat",
"time-progress": {
 "exp": {
  "left": { "op": "der", "var": "x" },
  "op": "=",
  "right": {
   "left": {
    "left": -0.2,
    "op": "*",
    "right": "x"
   },
   "op": "+",
   "right": 5.0
  }
 }
}
```



JANI: Guards for Transitions

```
"location": "l_heat",
"guard": {
 "exp": {
  "left": 22,
  "op": "≤",
  "right": "x"
 }
},
"destinations": [
 { "location": "l_cold" }
]
```



JANI: Example Specification

```
"properties": [
 {
  "expression": {
   "fun": "max",
   "op": "filter",
   "states": { "op": "initial" },
   "values": {
    "exp": {
     "left": true,
     "op": "U",
     "right": {
      "left": "x",
      "op": "≤",
      "right": 20.0
     },
     "time-bounds": {
      "upper": 20
     }
    },
   "op": "Pmax"
   }
  },
  "name": "specification"
 }
]
```

- Defines a reachability property
- Goal condition: $x \leq 20$
- Only in 20 time-steps
- RealySt checks whether this goal can be reached in any location

State JANI Parser in RealySt

- Supported rectangular hybrid automata before
- Linear hybrid automata were not supported
- Specifications were missing while creating a JANI-file
- Resets were not handled

Implementation – Structure of a Linear Expression

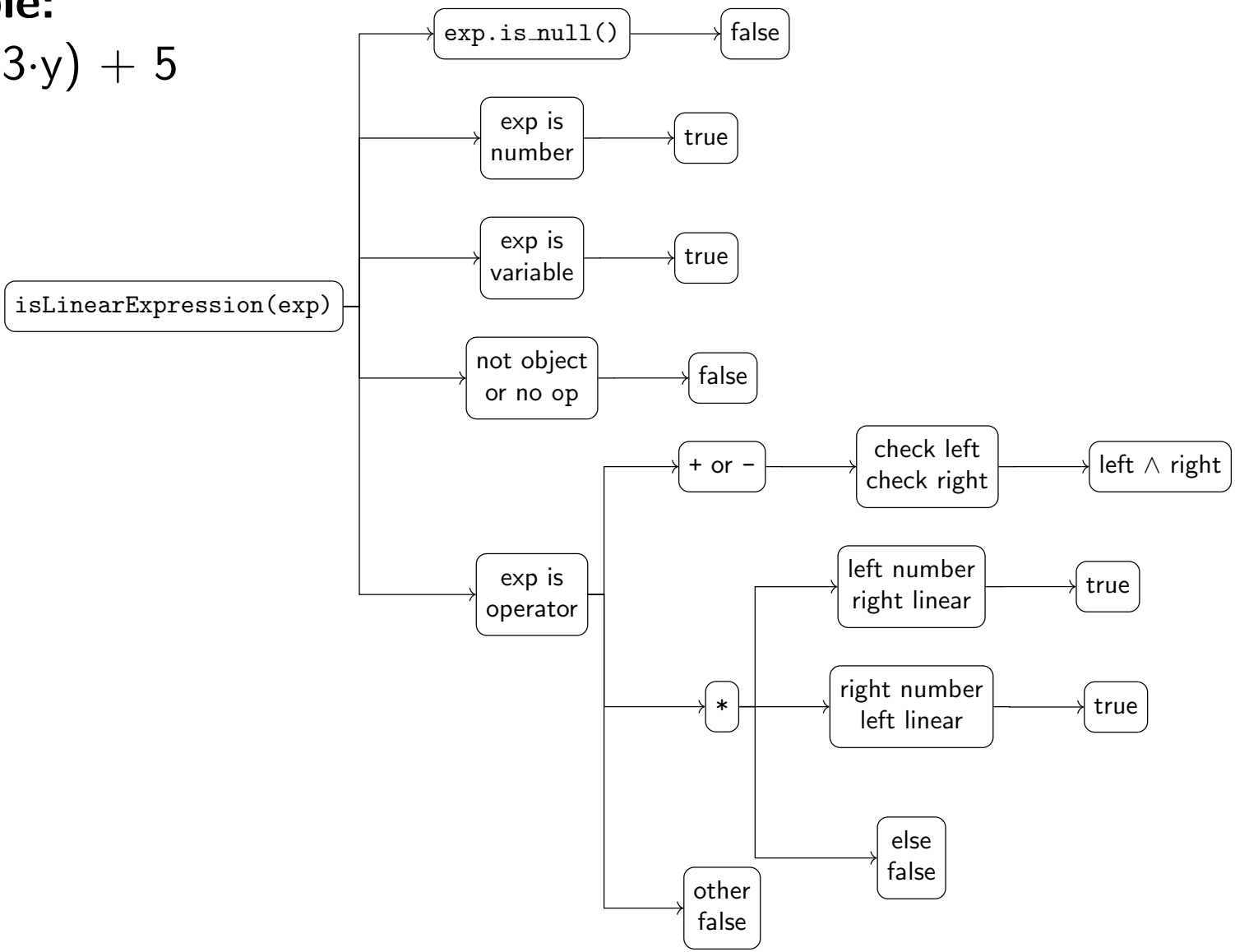
```
{
 "op": "+",
 "left": {
  "op": "+",
  "left": {
   "op": "*",
   "left": 2,
   "right": "x"
  },
  "right": {
   "op": "*",
   "left": 3,
   "right": "y"
  }
 },
 "right": 5
}
```

- $(2 \cdot x + 3 \cdot y) + 5$
- JANI stores expressions as trees
- Inner nodes are operators
- Leaves are variables or constants

Implementation – Detecting Linear Expression

Example:

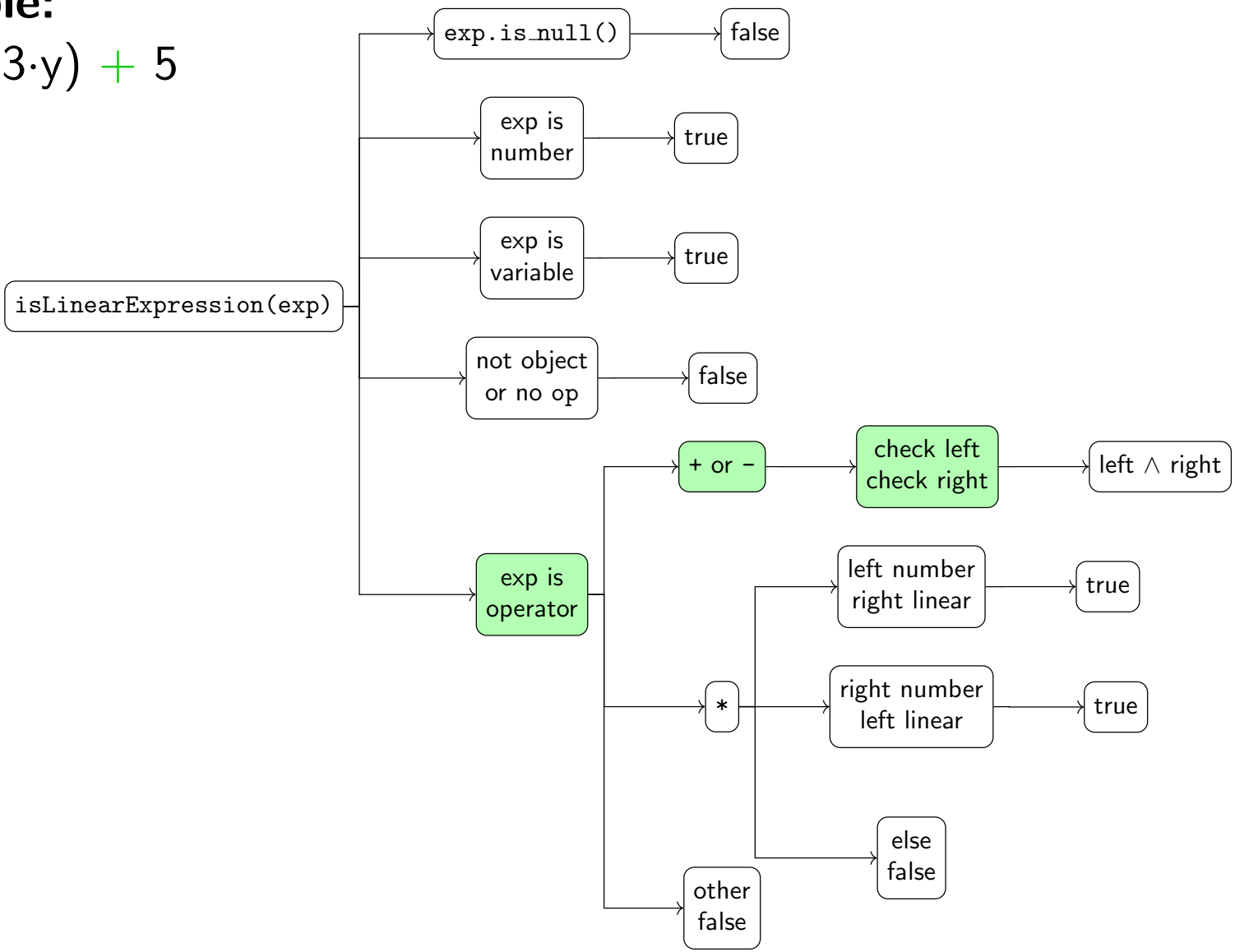
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

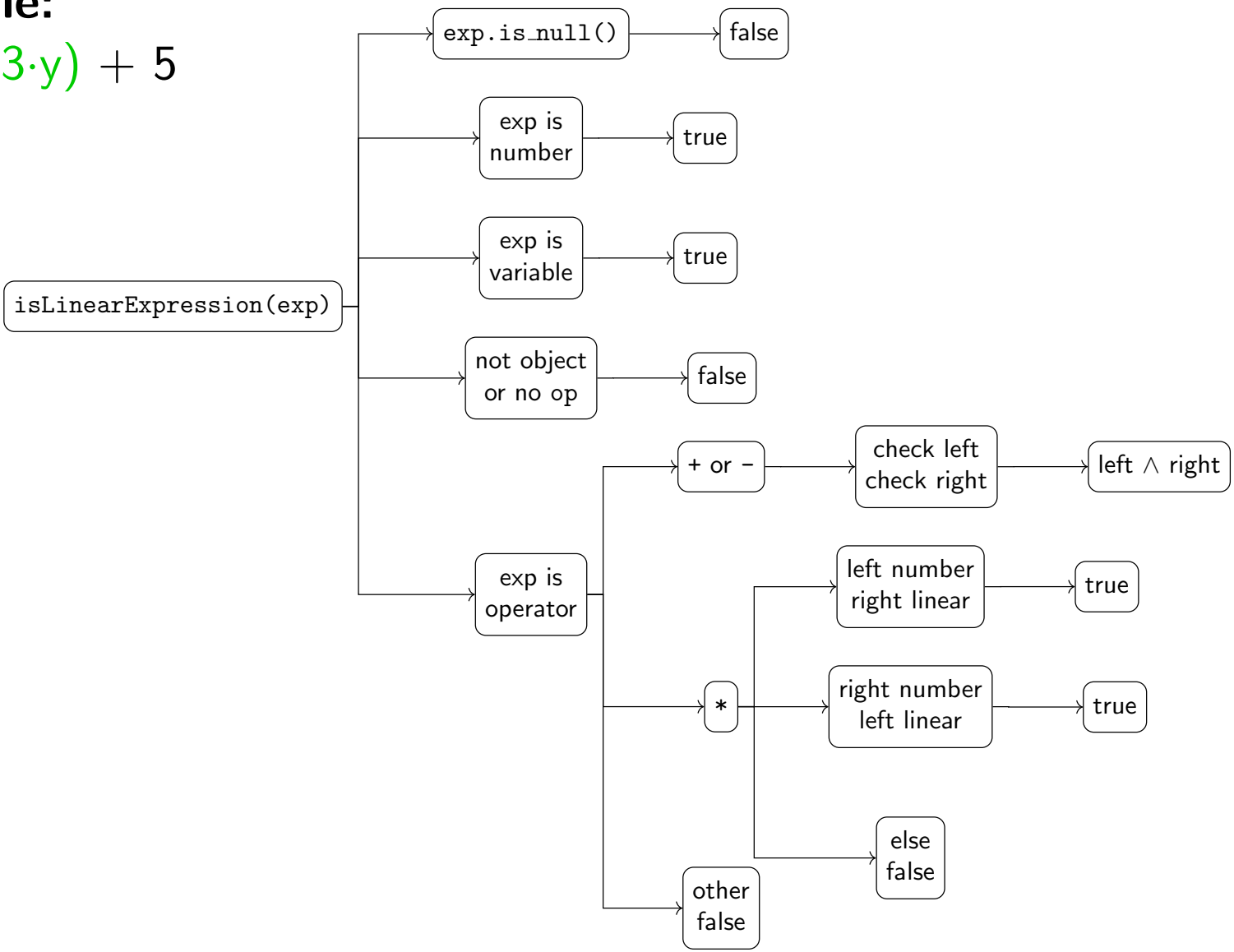
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

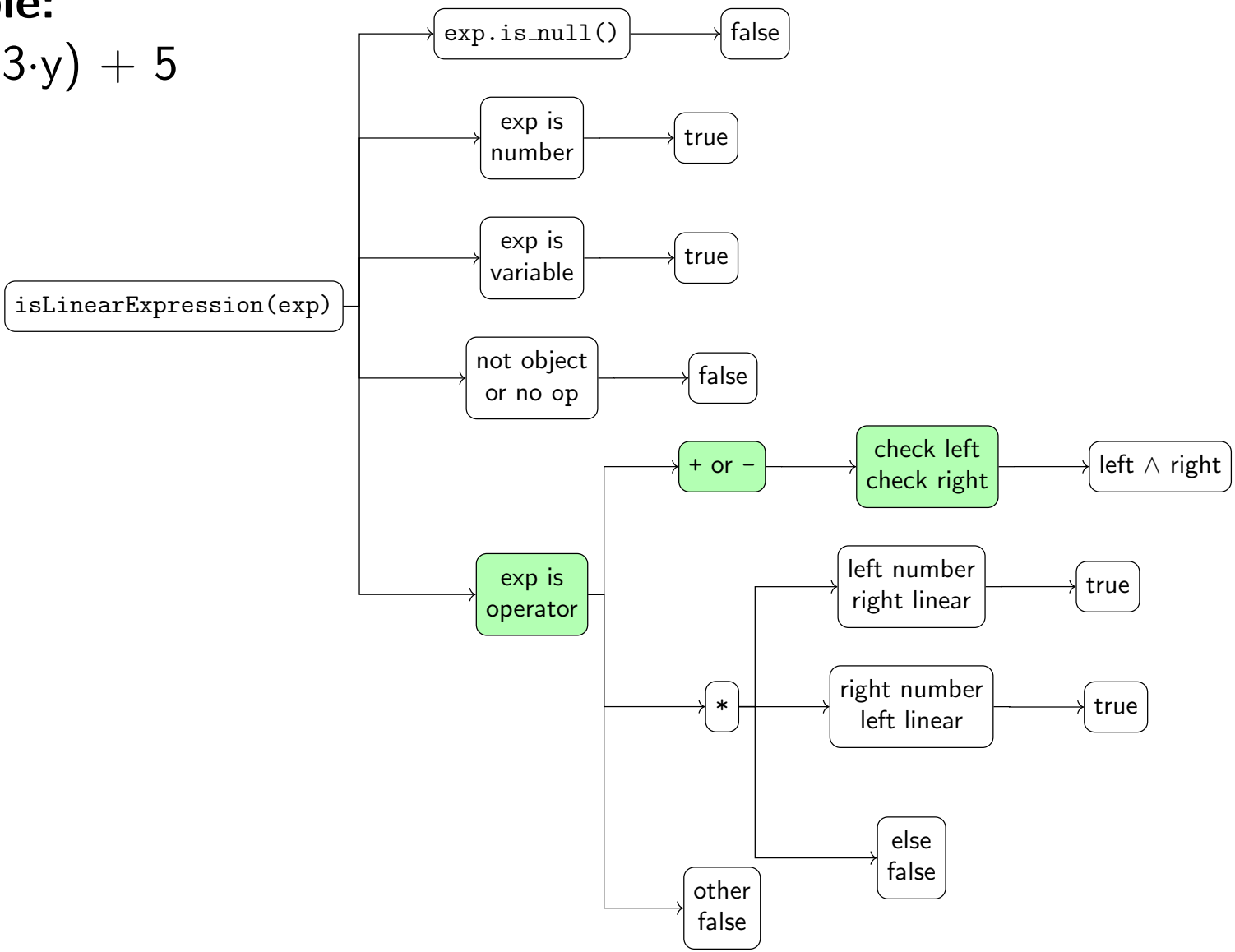
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

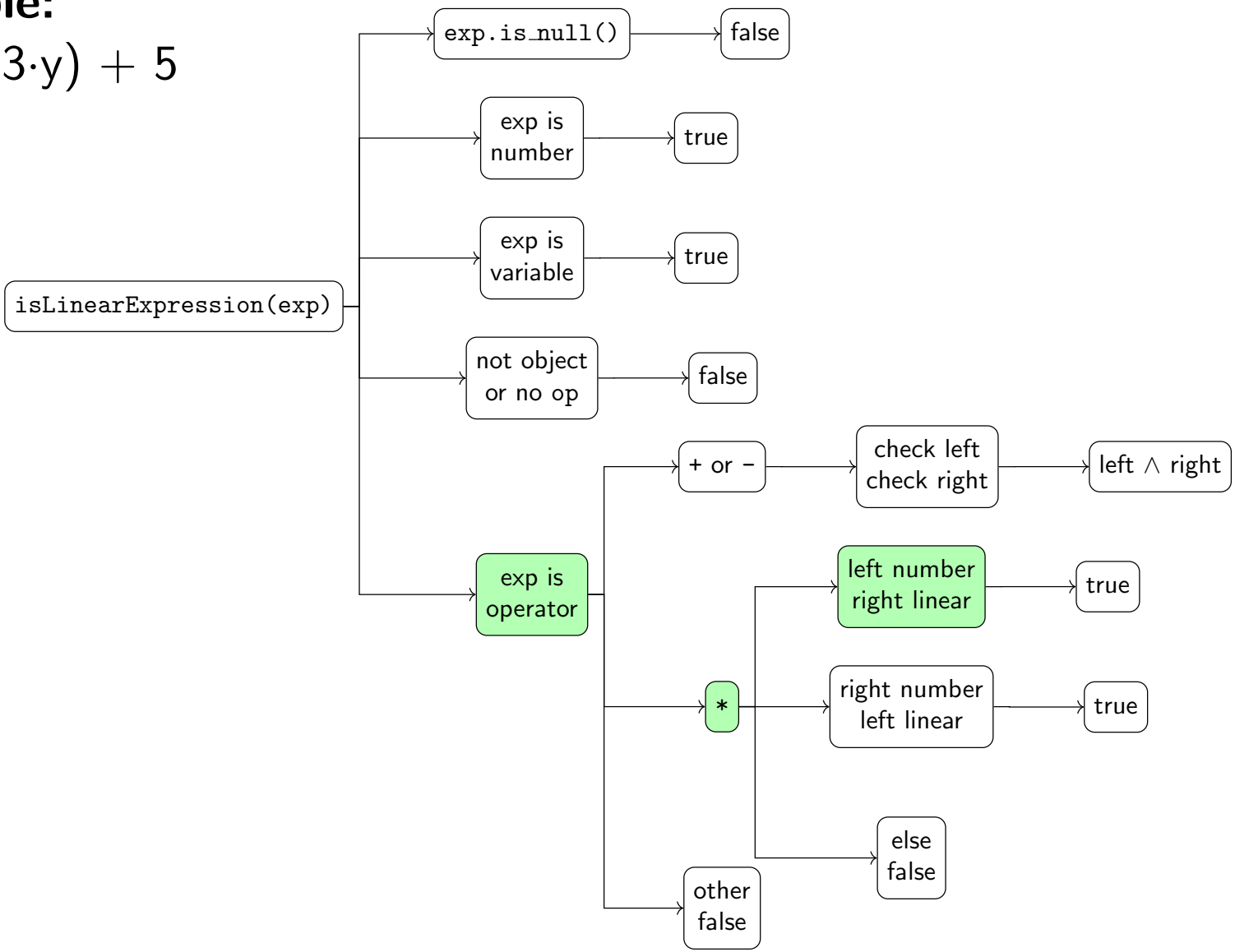
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

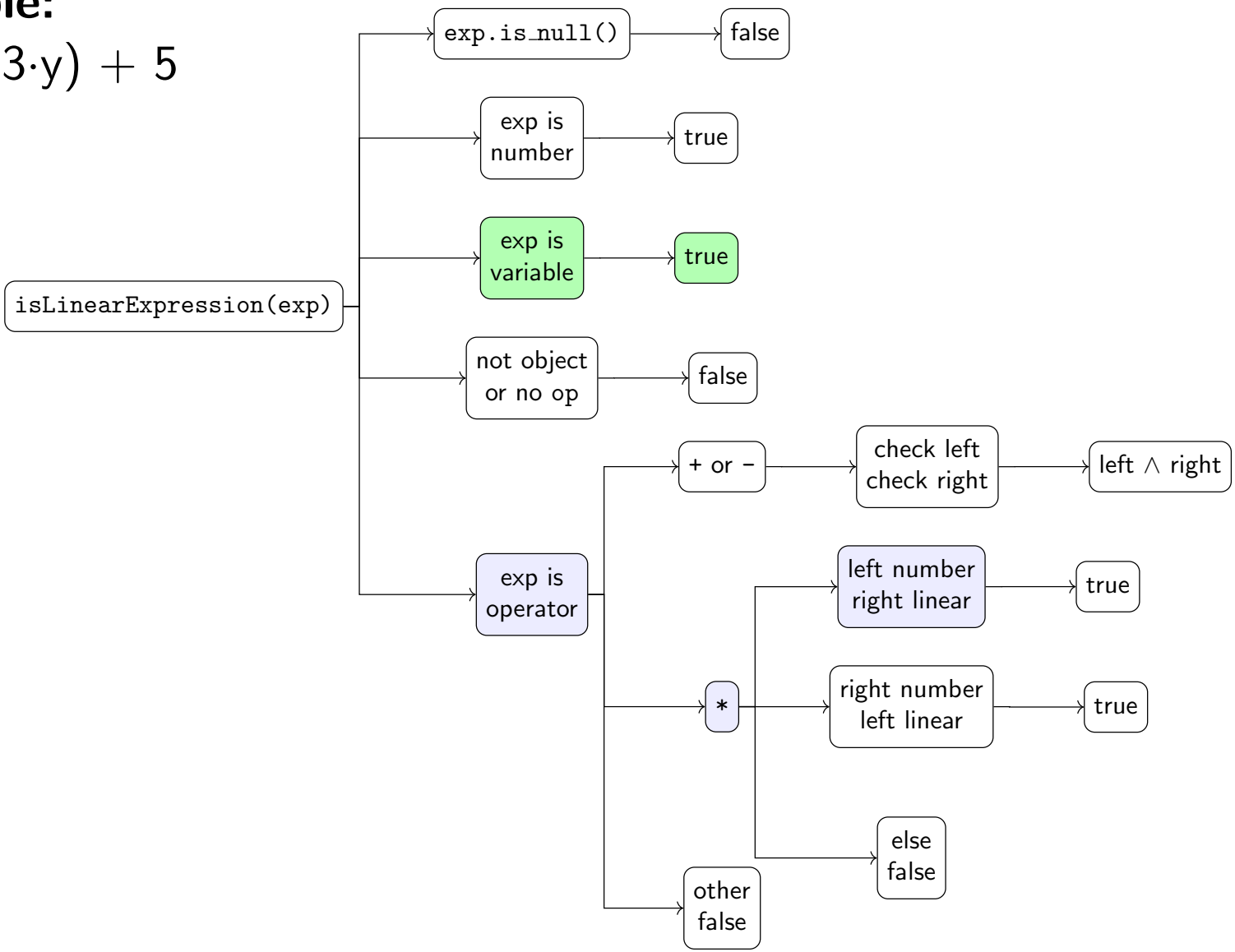
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

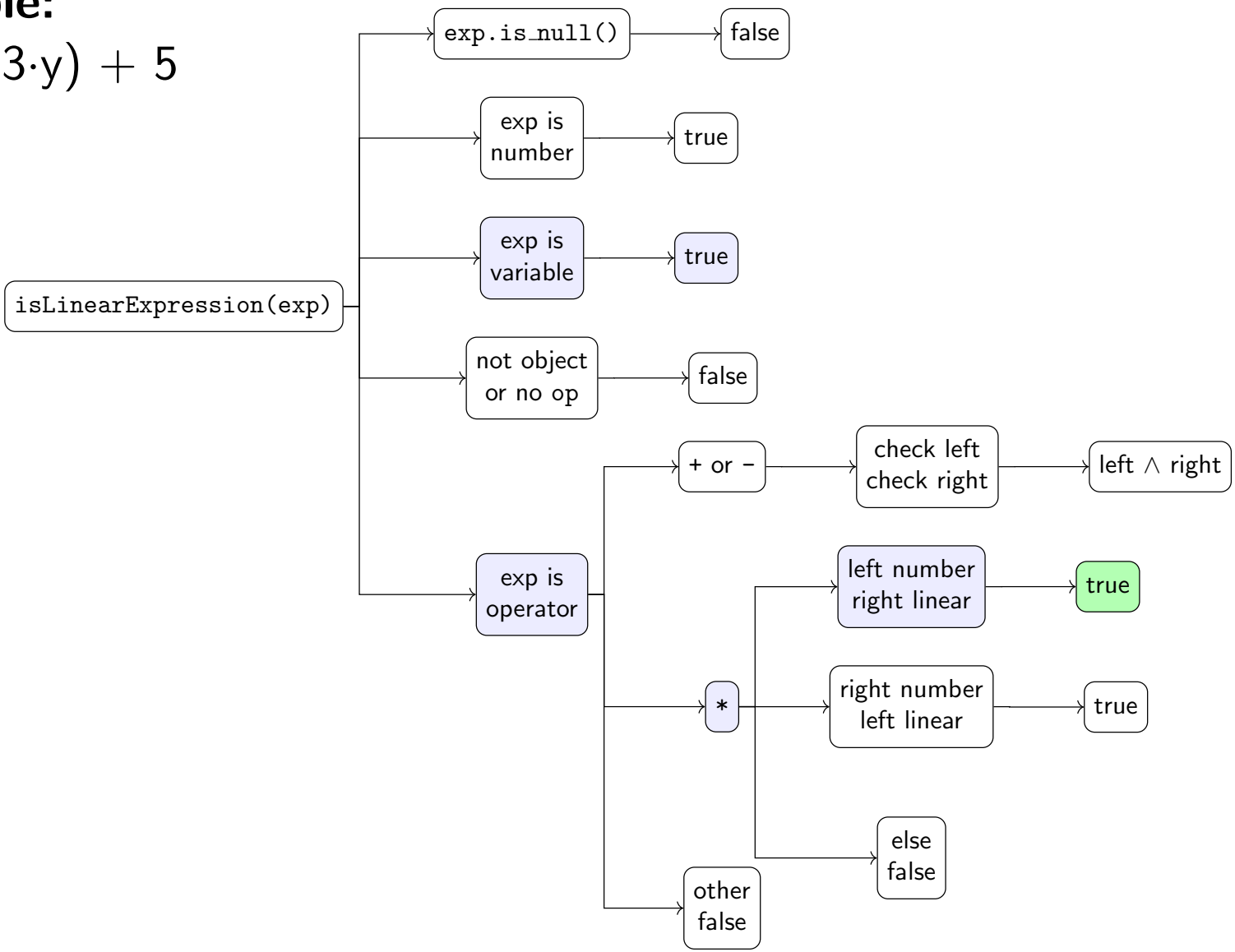
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

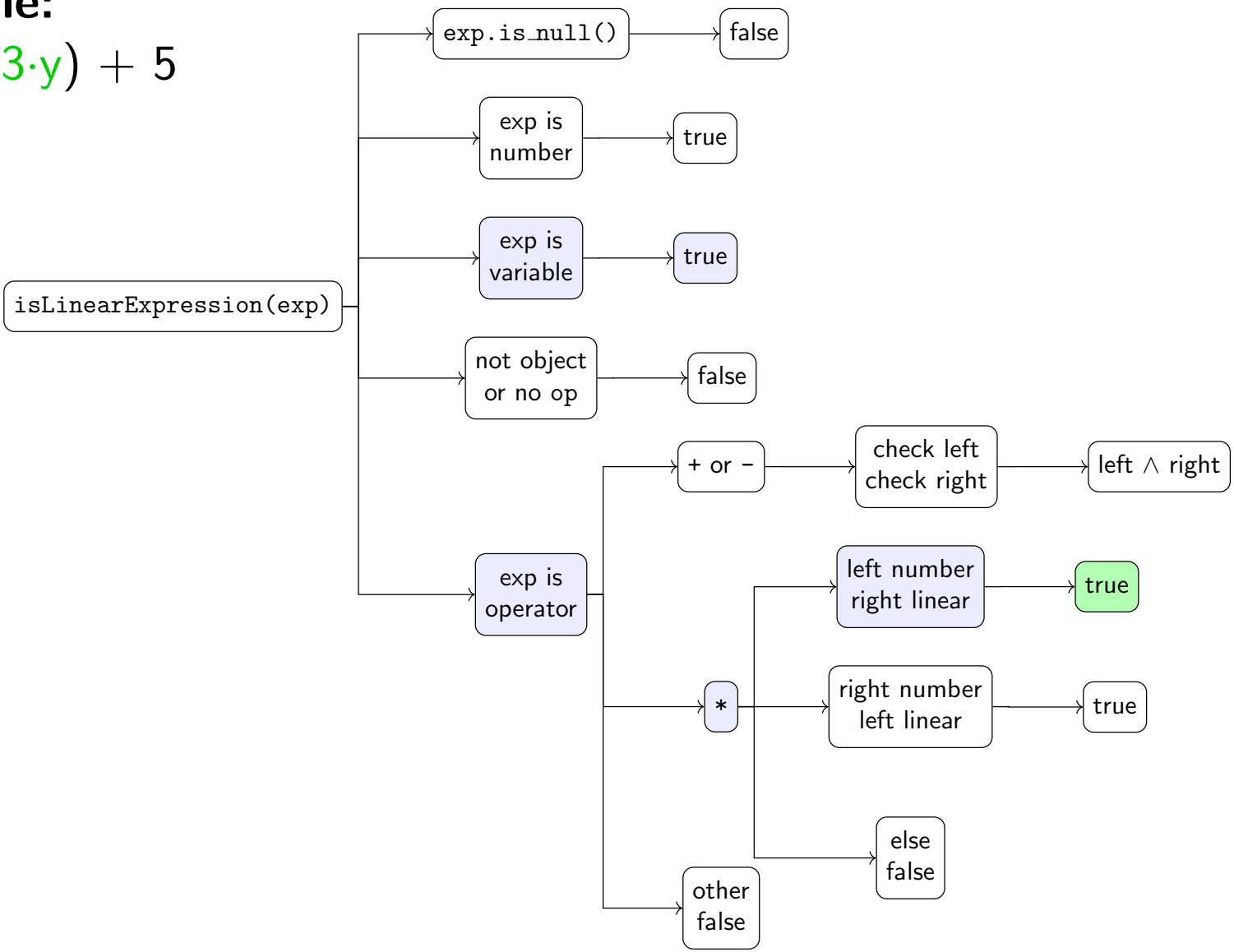
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

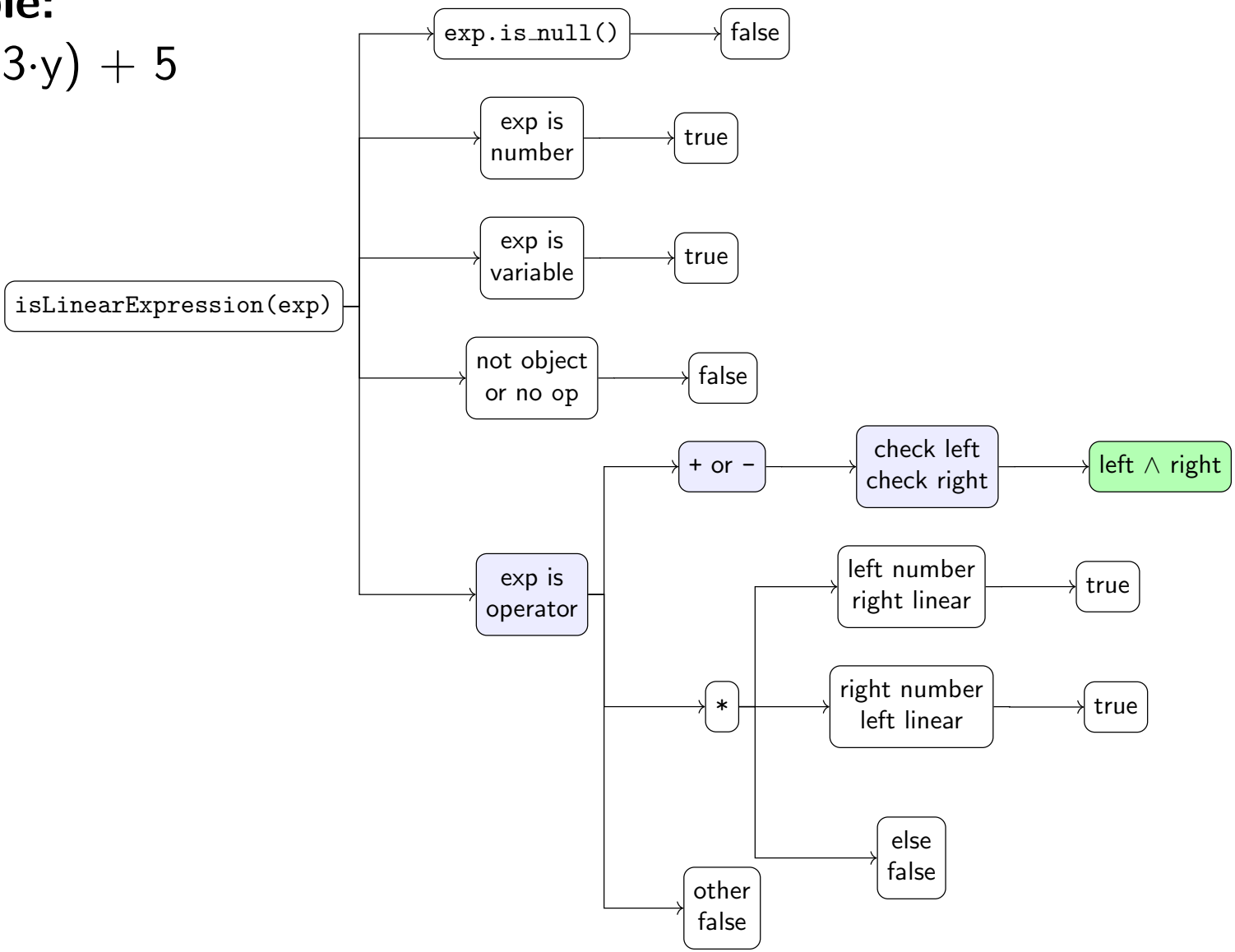
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

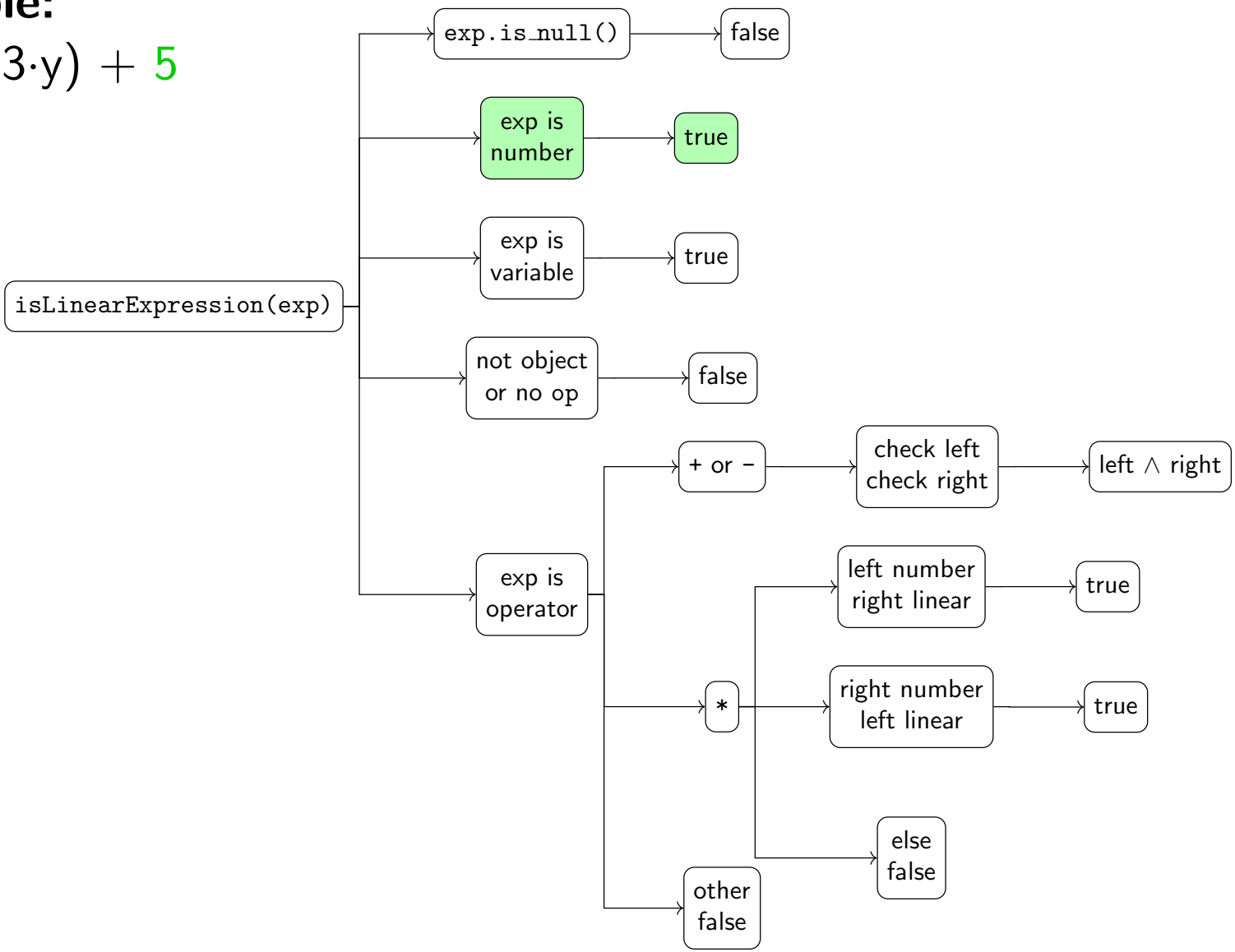
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

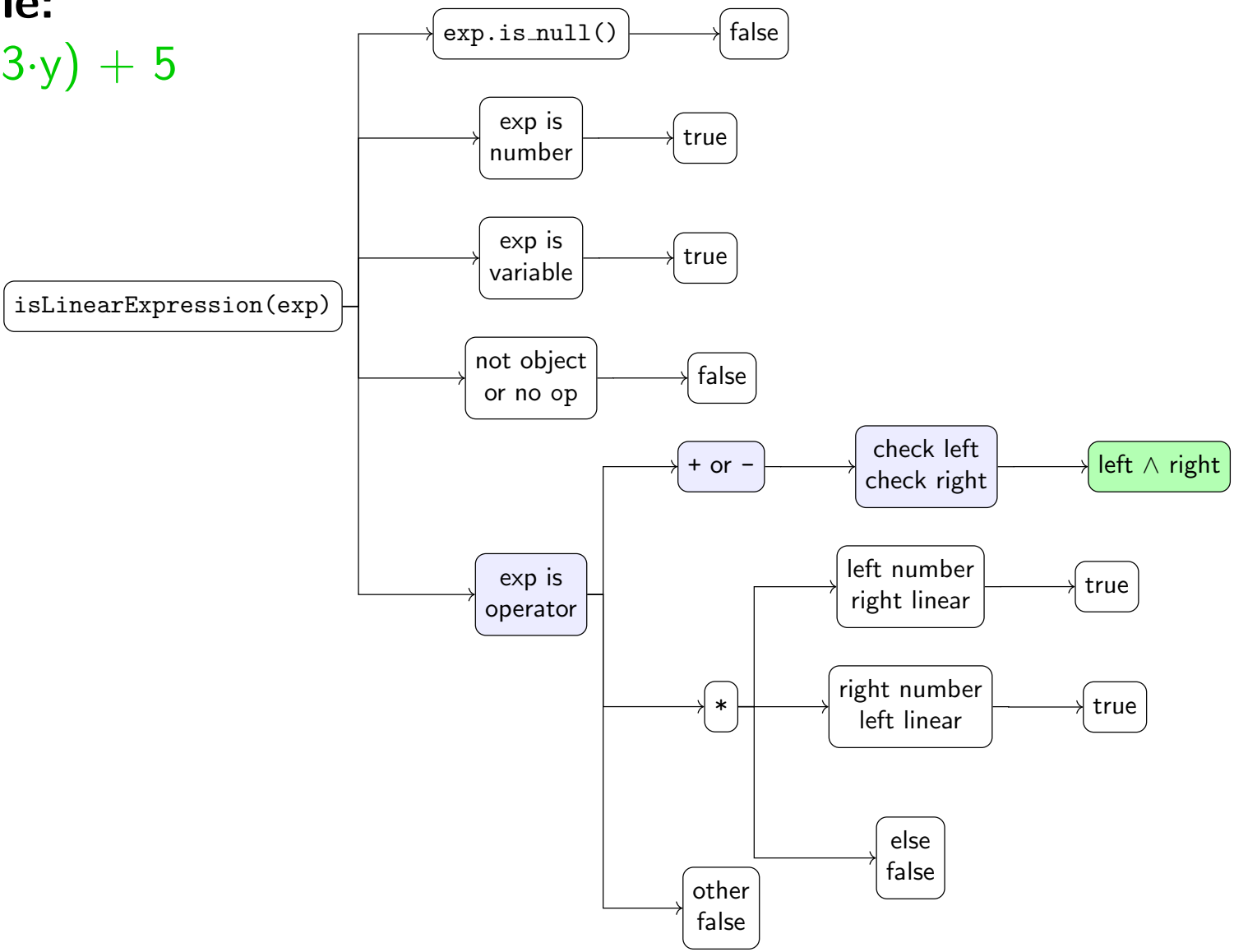
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Detecting Linear Expression

Example:

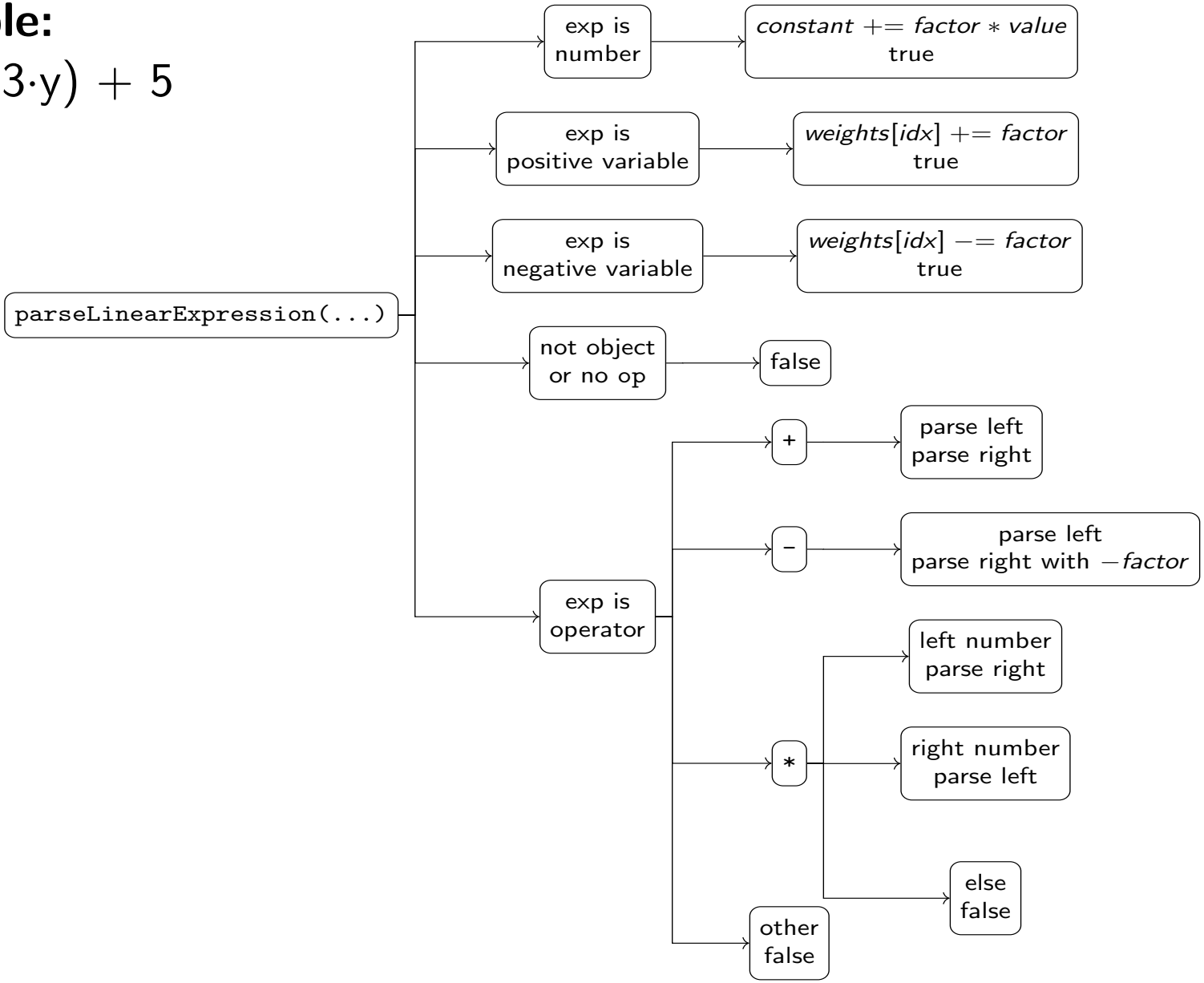
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

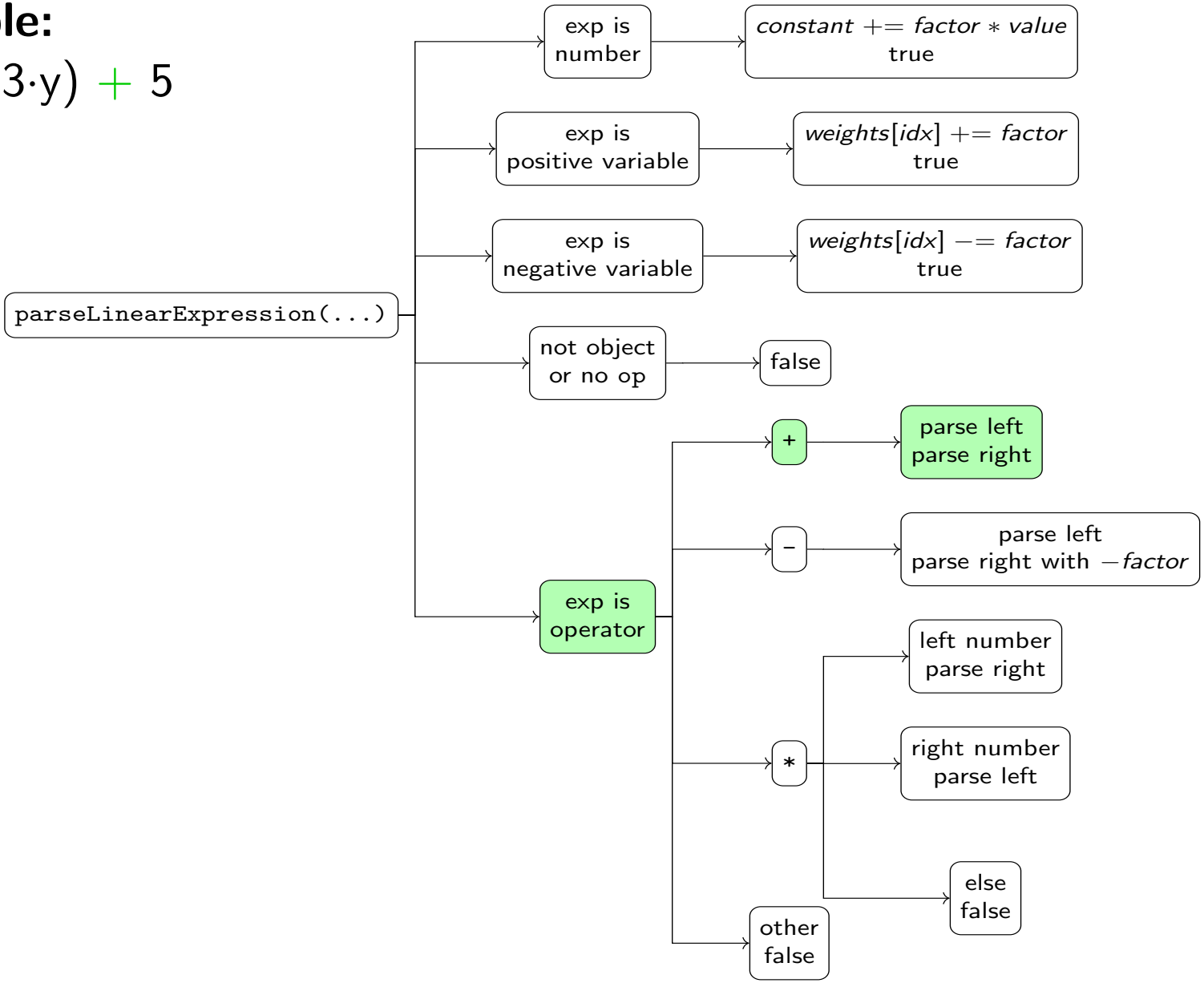
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

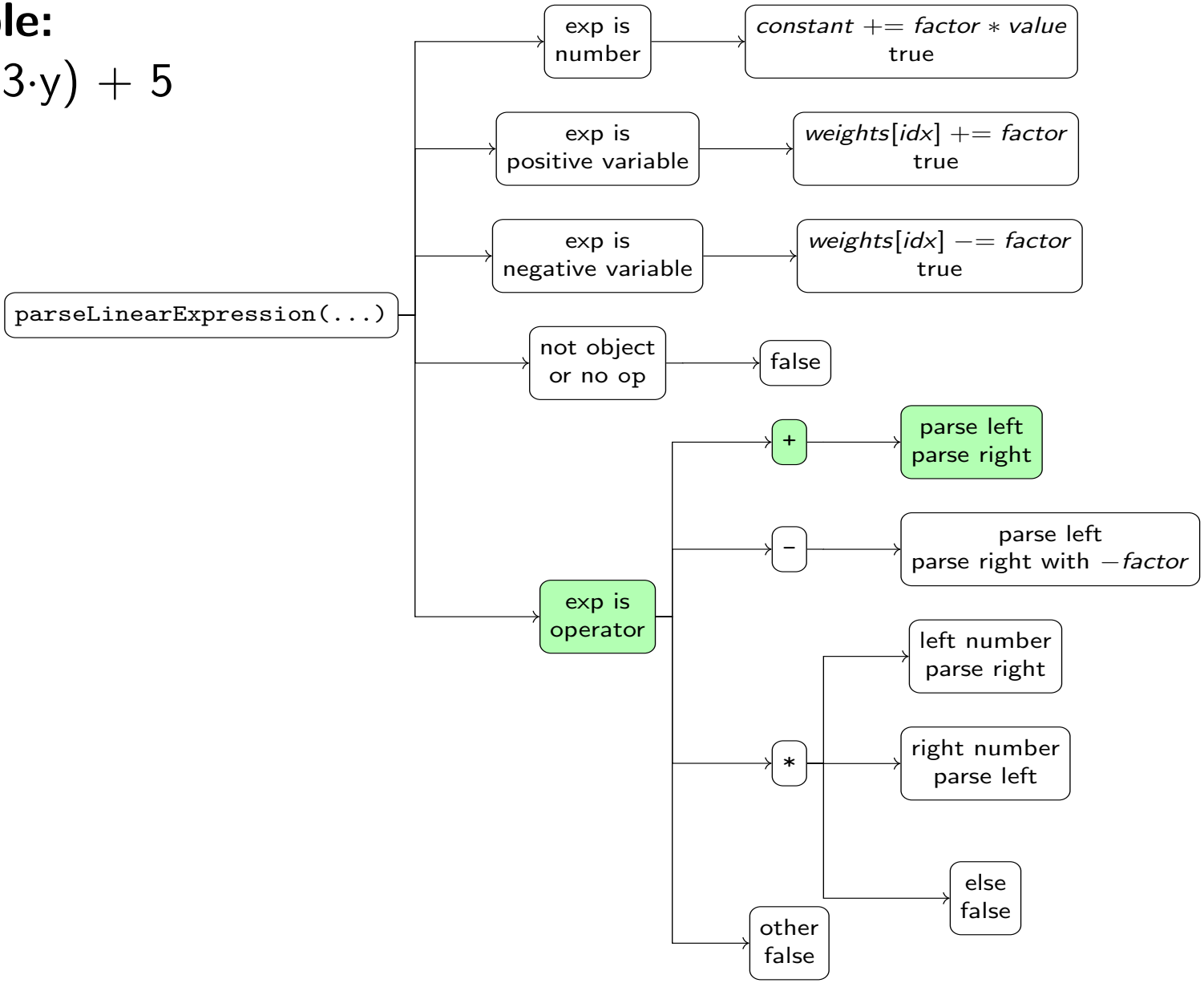
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

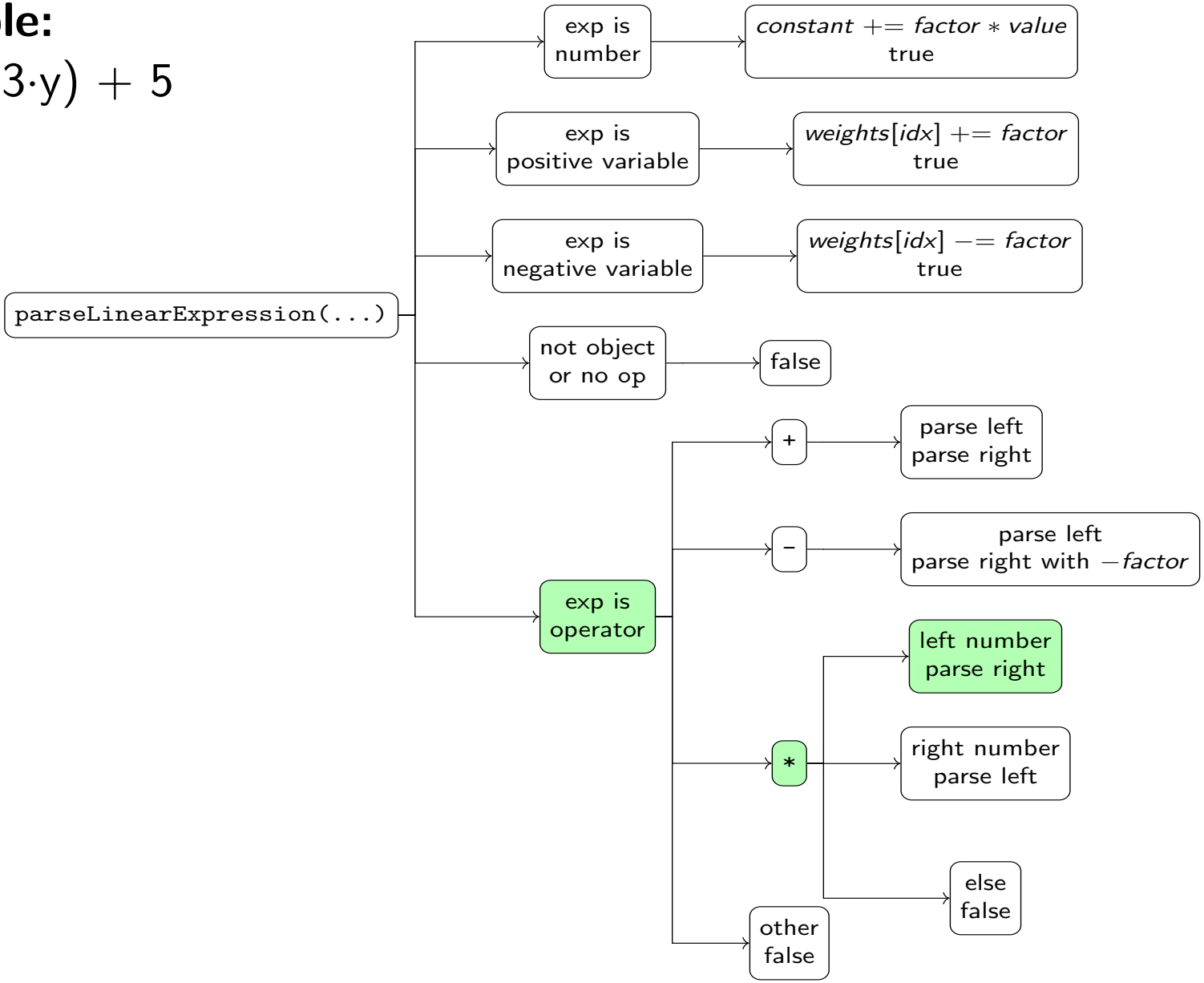
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

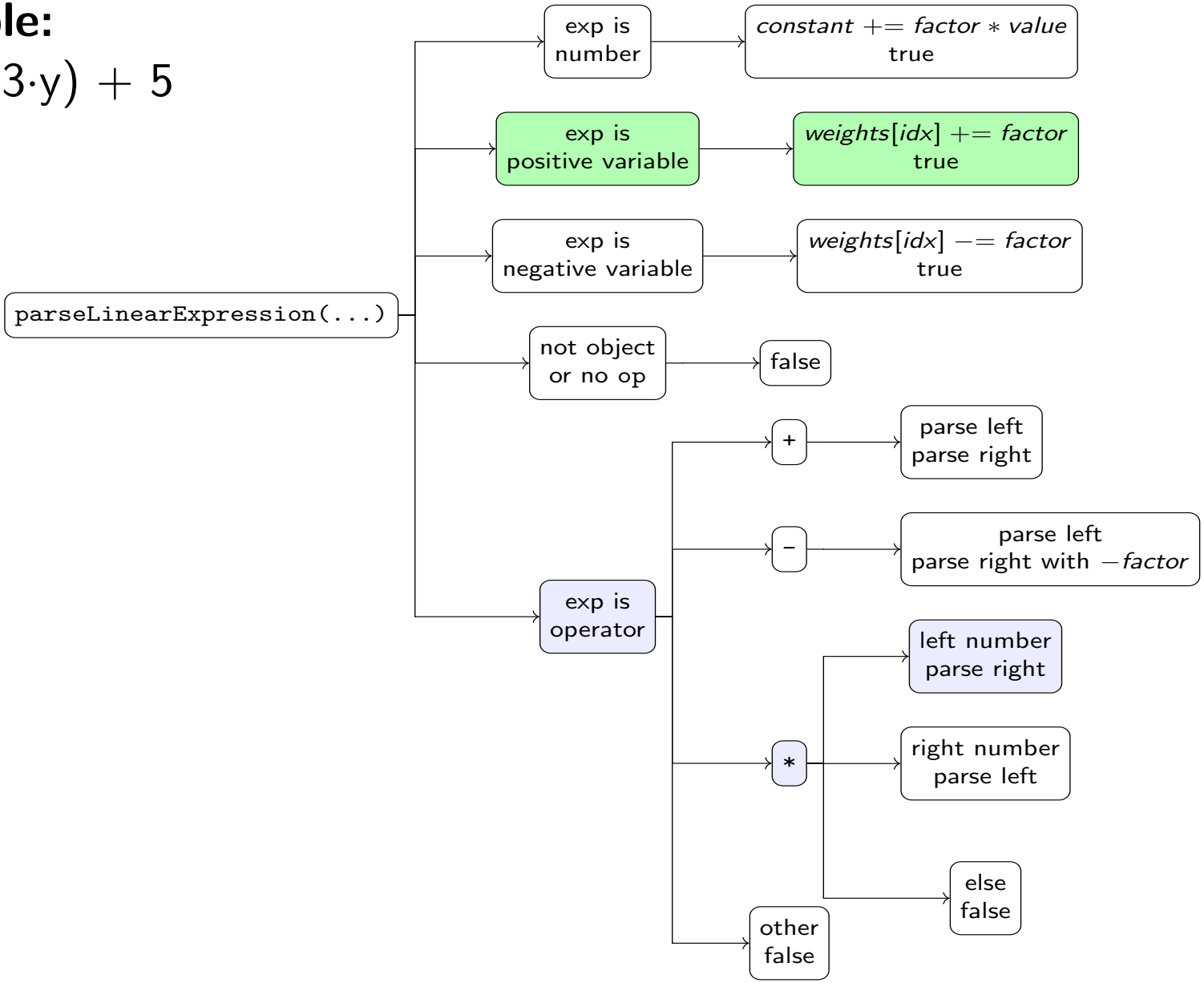
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

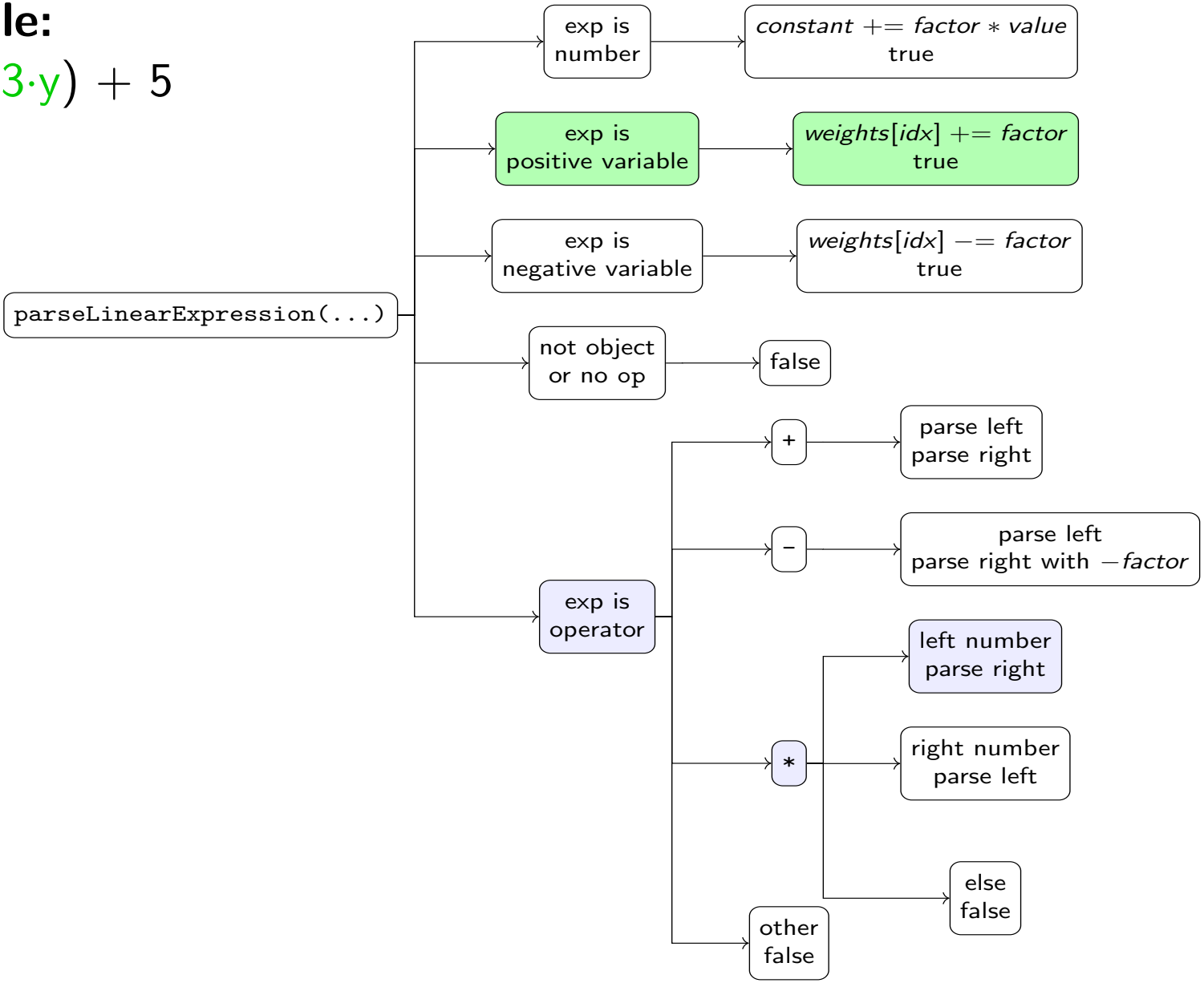
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

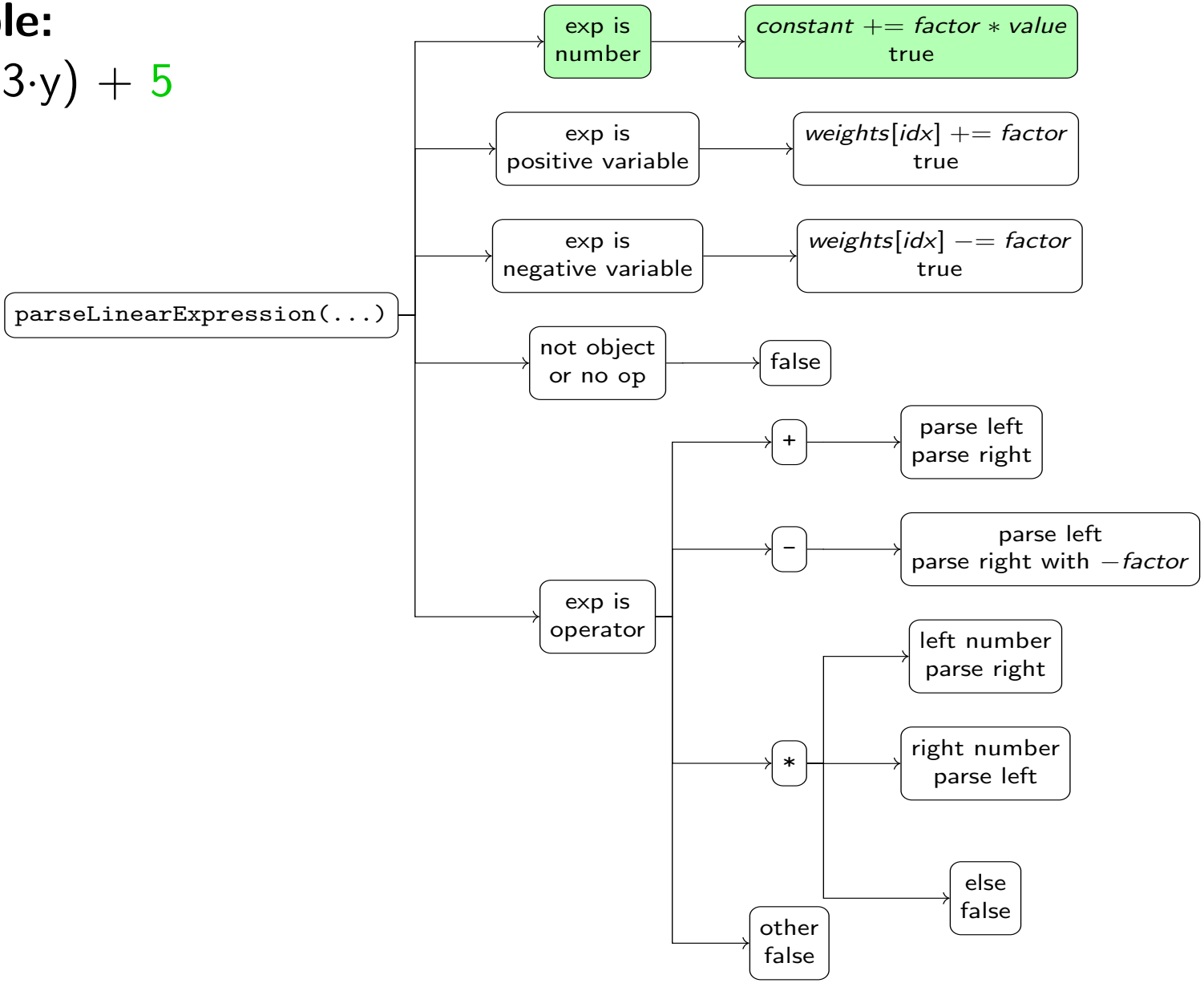
$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Parsing Linear Expression

Example:

$(2 \cdot x + 3 \cdot y) + 5$



Implementation – Saving Linear Expressions

Data Types

- `using WeightedVariables = std::map<VariableIndex, Number>;`
- `struct LinearFlow { LinearExpression linearTerms; Number constant = 0; }`
- `using GeneralFlowMap = std::variant<carl::Interval<Number>, LinearFlow>;`

Example

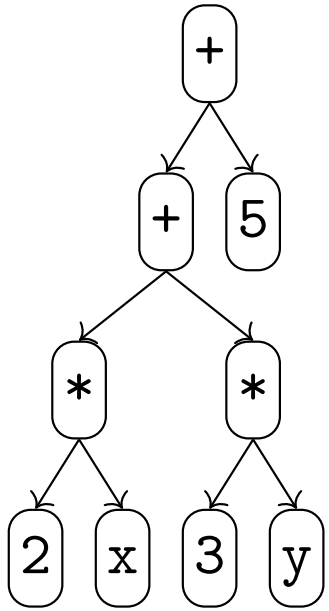
- `WeightedVariables: weights[x] = 2; weights[y] = 3;`
- `LinearFlow: flow.linearTerms = {{2,x},{3,y}}; flow.constant = 5;`
- `GeneralFlowMap: flows[z] = flow;`

Implementation – Linear Attributes

- Functions to detect and parse linear terms
- Extensions in `parseFlow()`, `parseGuards()`, `parseInvariant()` and functions for specifications
- New maps for each attribute with same structure
- `--LTIAnalysis` relevant for parseFlow()

Implementation – Creating Linear Terms

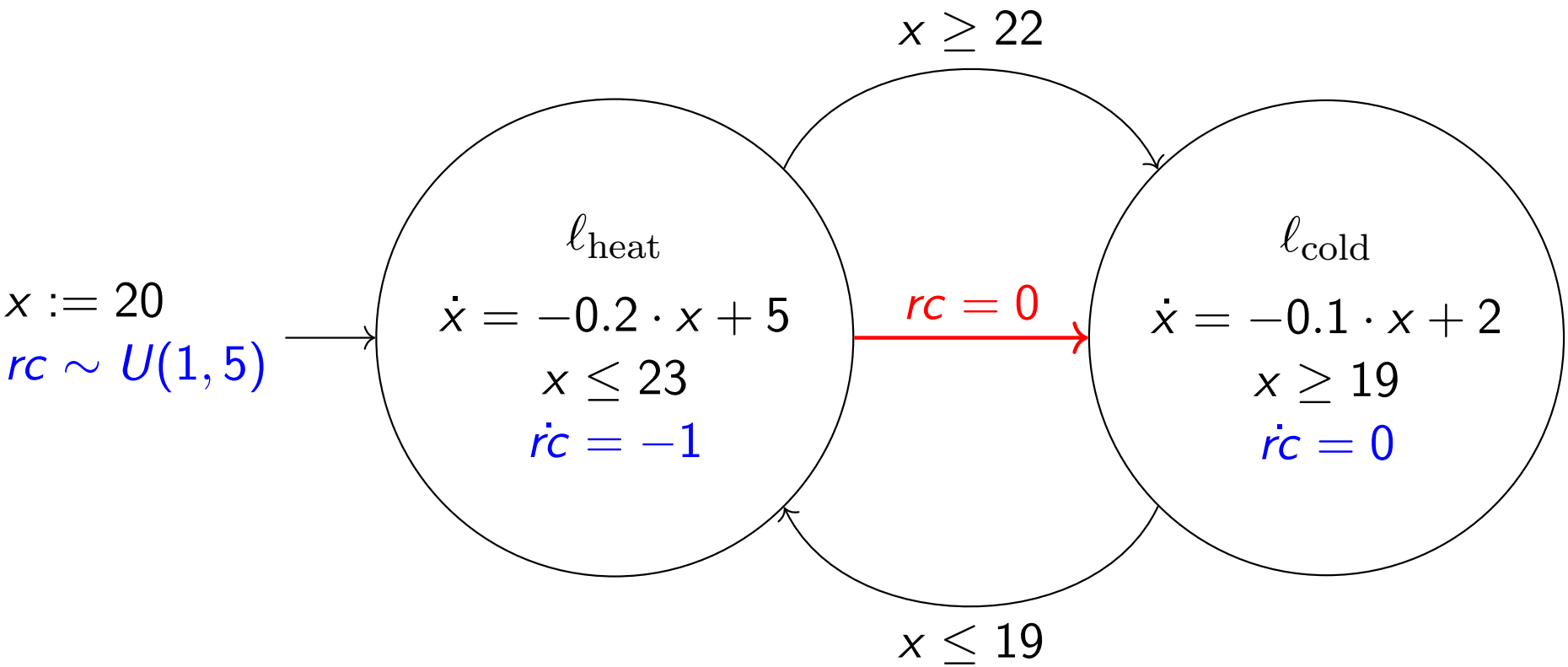
- Input: weighted variables; optional constant
- Example: $(2 \cdot x + 3 \cdot y) + 5$
- Construction process
  - create multiplication nodes $2 \cdot x$ and $3 \cdot y$
  - combine terms with $+$
  - append constant 5 with $+$ separately



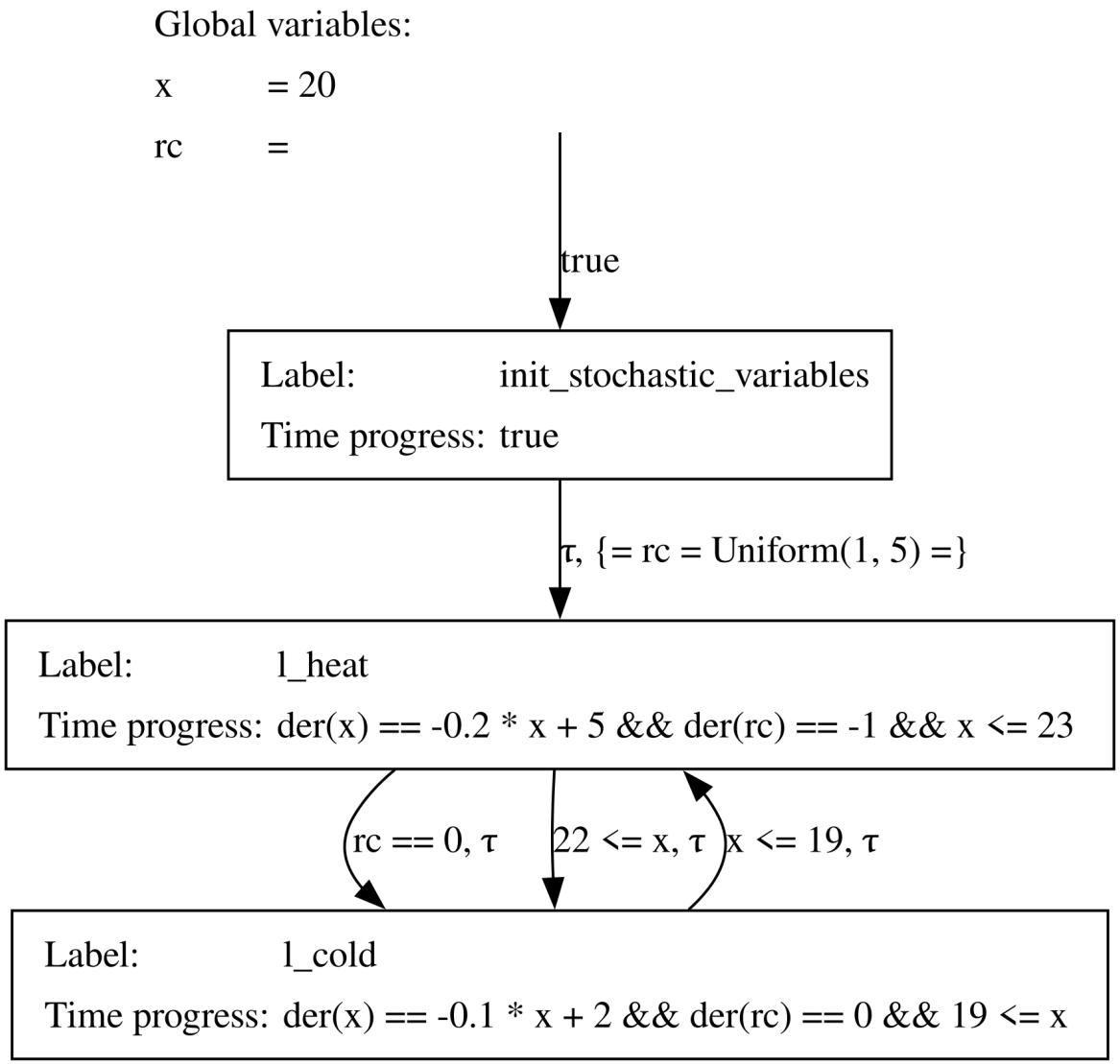
Implementation – Extension for other functionalities

- Specifications
  - Were already handled during the import
  - Added time-bounds in both directions
  - Extended for creating JANI-files
  - Support for defining goal locations for REALYST
- Resets
  - Resets were not handled in general
  - Only added support for linear resets

Plotting with Modest



Plotting with Modest



Results

- Linear Hybrid Automata can be parsed in RealySt
- Support for resets in general added
- Specifications included while creating Jani-files
- Approach of defining goal locations only compatible with RealySt
- Disjunction of specifications tested with Jani-files

Future Work

- Improve compatibility for goal locations in specifications
- Handling the $\vee$-operator
- Improve handling for multiple automata
- Further testing the parser with other tools