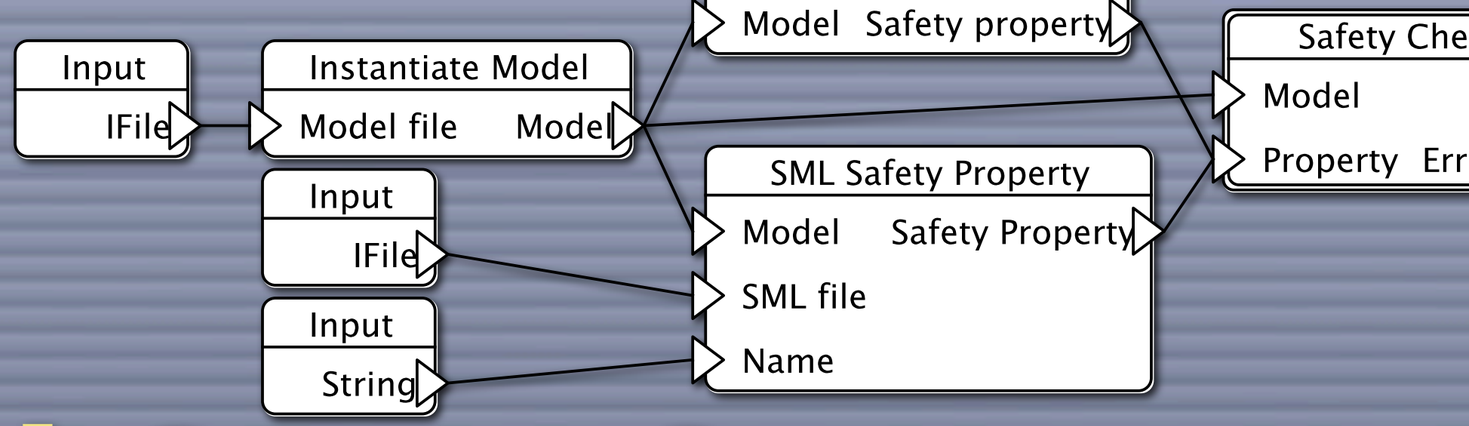
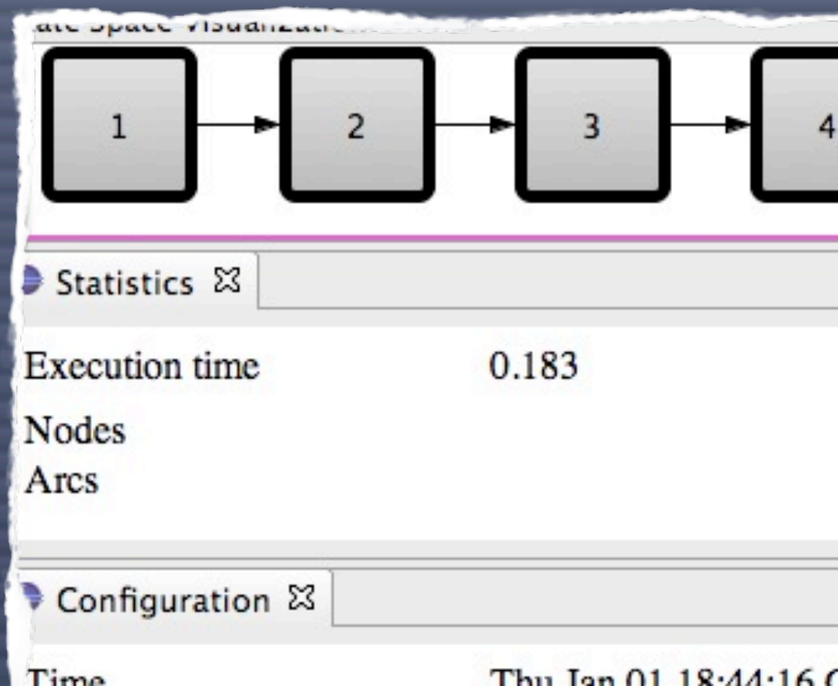


Advanced State Space Methods and ASAP: Practical Use

Michael Westergaard
Department of Computer Science
Aarhus University
mw@cs.au.dk

$V := \{ s_0 \}$
$W := \{ s_0 \}$
while $W \neq \emptyset$ **do**
Select an $s \in W$
$W := W \setminus \{ s \}$
if $\neg I(s)$ **then**
    return false
for all t, s' **such that** $s \rightarrow^t s'$ **do**
    if $s' \notin V$ **then**
        $V := V \cup \{ s' \}$
        $W := W \cup \{ s' \}$
return true

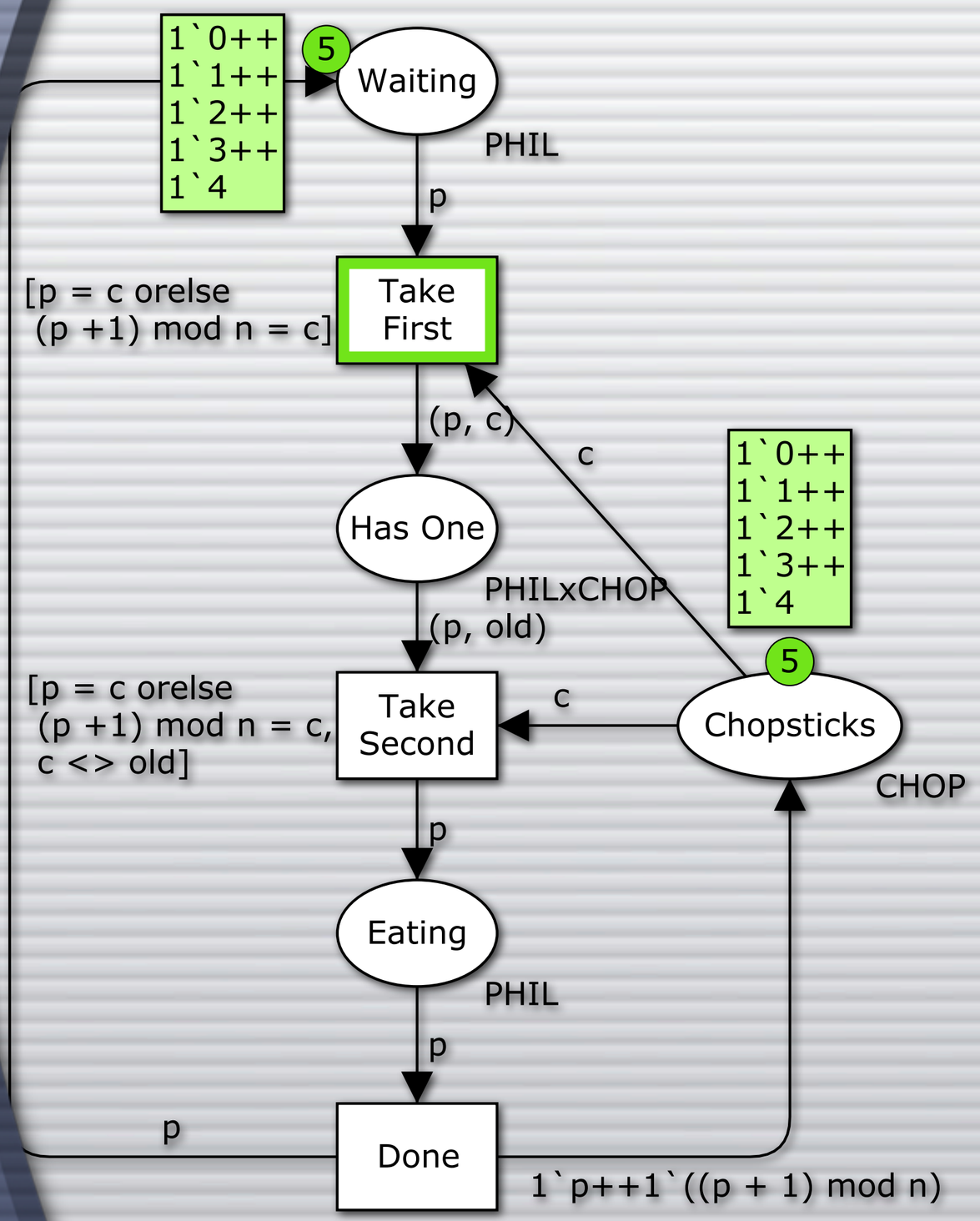


Verification Jobs

- Verification of a a model is done using a verification project consisting of
  - CPN **Models** to be analyzed
  - **Queries** expressing the properties we are interested in
  - **Verification jobs** coupling models, queries, and state space methods
  - **Reports** reflecting results of executing verification jobs

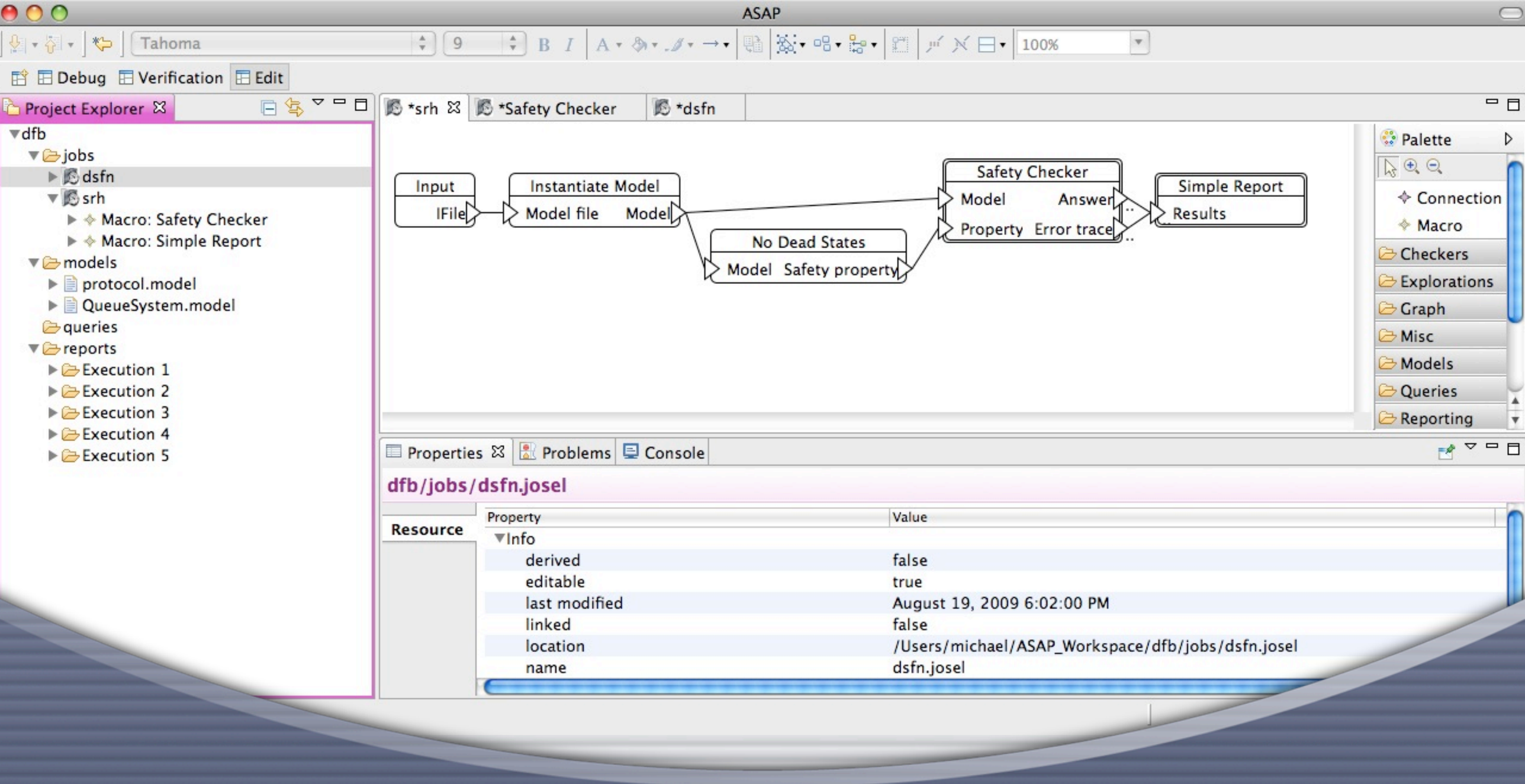
Example: Dining Philosophers

- Simple execution of the model



Demo: Dining Philosophers (01)

- Do a bit of simple simulation



Example: Check for Deadlocks

Demo: Check for Deadlocks (02)

- Creation of Verification project
- Loading models
- Creating a Verification job from a template
- Executing a job template
- Reporting

ASAP

Verification | Edit

Project Explorer

- demo
  - jobs
    - checker
      - Macro: Safety Checker
      - Macro: Simple Report
  - models
    - deadlocking philosophers.mod
      - Declarations
      - New Page
  - queries
  - reports
    - Execution 1
      - Configuration.rptdocument
      - Results.rptdocument
      - Statistics.rptdocument

Progress

No operations to display at this time.

checker | Configuration | Statistics | Results

No Dead States	false
Error trace	New_Page.Has_One: 1`(0,0) ++ 1`(1,1) ++ 1`(2,2) ++ 1`(3,3) ++ 1`(4,4) New_Page.Philosophers: 1`5
Error trace	New_Page.Has_One: 1`(0,1) ++ 1`(1,2) ++ 1`(2,3) ++ 1`(3,4) ++ 1`(4,0) New_Page.Philosophers: 1`5

Console

Simulator Console

```
- let open JavaExecute in
case (SafetyChecker.explore true 0 (CPN'Structure'MLExplicitRemoveStorage'4.emptyStorage { init_size = 0 }()) (
of [] => execute "result" []
 | _ => ()
end
val it = () : unit
-
```

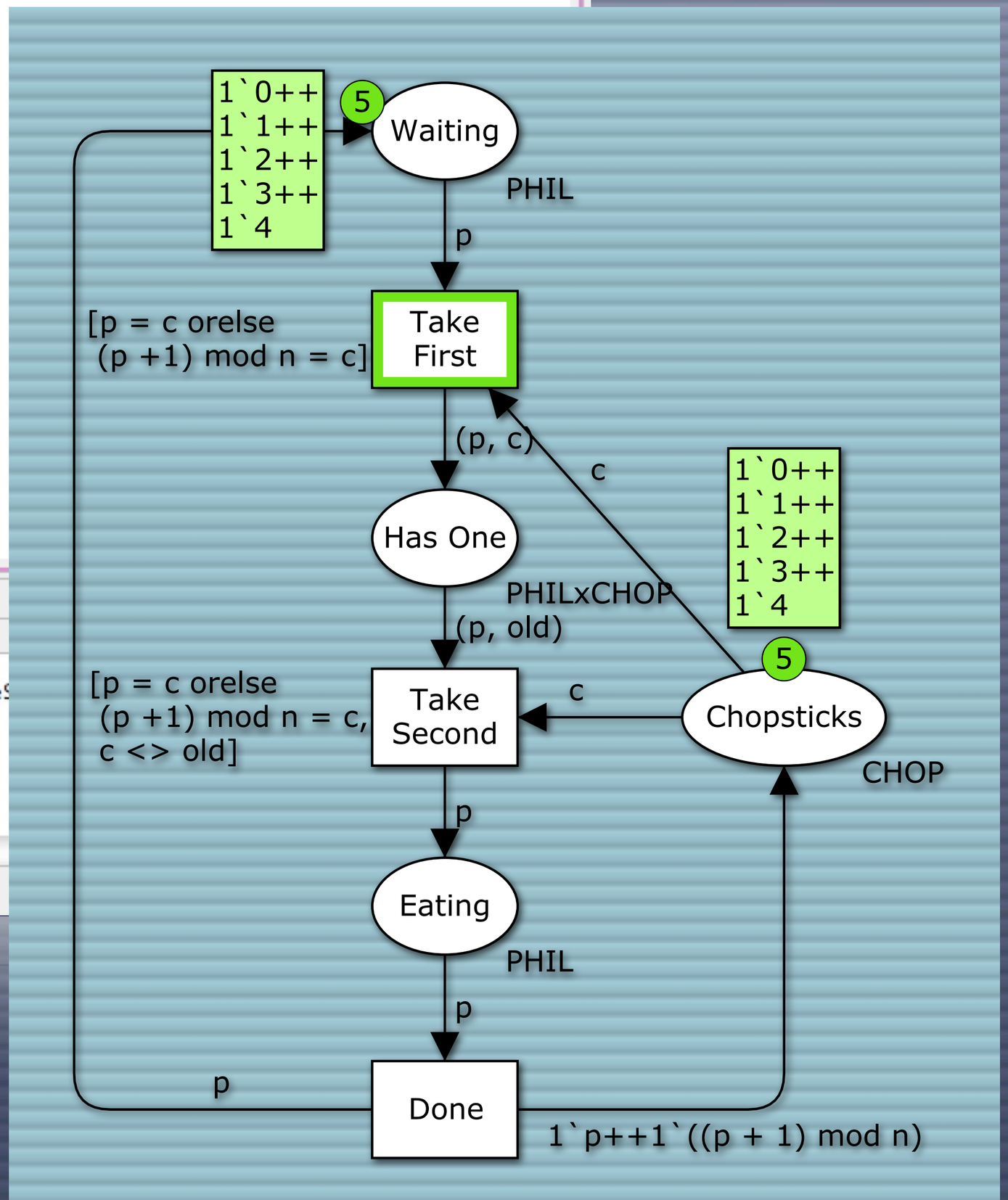
checker | Configuration | Statistics | Results

No Dead States	false
Error trace	New_Page.Has_One: 1`(0,0) ++ 1`(1,1) ++ 1`(2,2) ++ 1`(3,3) ++ 1`(4,4) New_Page.Philosophers: 1`5
Error trace	New_Page.Has_One: 1`(0,1) ++ 1`(1,2) ++ 1`(2,3) ++ 1`(3,4) ++ 1`(4,0) New_Page.Philosophers: 1`5

Console

Simulator Console

```
- let open JavaExecute in
case (SafetyChecker.explore true 0 (CPN'Structure'MLExplicitRemoveS
of [] => execute "result" []
 | _ => ()
end
val it = () : unit
-
```



JoSEL: Background

- ASAP Supports a wide range of state space methods
  - Depth-first and breadth-first traversal
  - On-line and off-line analysis
  - Bit-state hashing and hash compaction
  - Sweep-line and ComBack methods
  - Safety properties, CTL, LTL

JoSEL: Background

- Applying a state space methods consists of
  1. Specifying a model to analyze
  2. Making queries expressing desired properties
  3. Select method to use for verification
  4. Set parameters of and instantiate the selected method
  5. Execute the traversal
  6. Post-process and interpret the results

JoSEL: Aim

- Develop a high-level language making it possible to tie the model, queries and desired state space method together
- Support research, education and industrial application scenarios

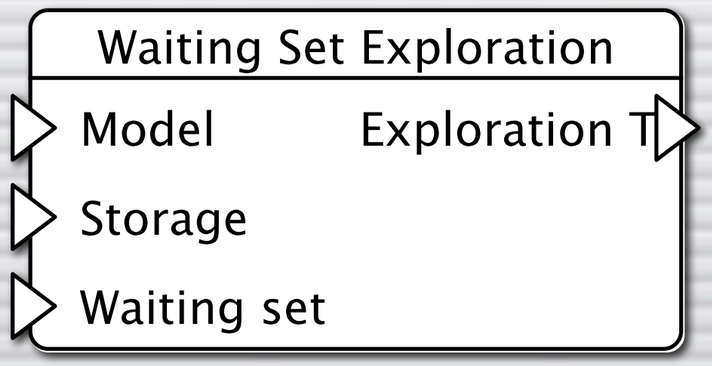
JoSEL: Requirements

- **Abstraction**: Hide details from users
- **Low-level control**: Make it possible to access details when required for performance

  The hash function used to hash states when storing in a hash table
- **Modularity**: Facilitate construction and use of building blocks (**templates**) in verification jobs
- **Extendibility**: Allow extension for new methods as needed

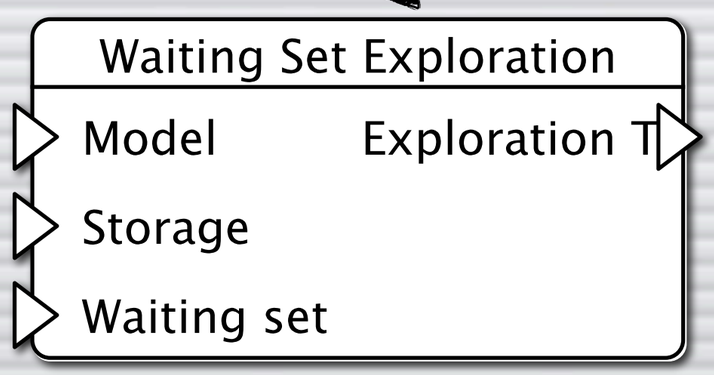
JoSEL Overview

- Graphical language inspired by data-flow diagrams and hierarchy of CP-nets
- Basic unit is a **task**
- Tasks have typed input and output **ports**



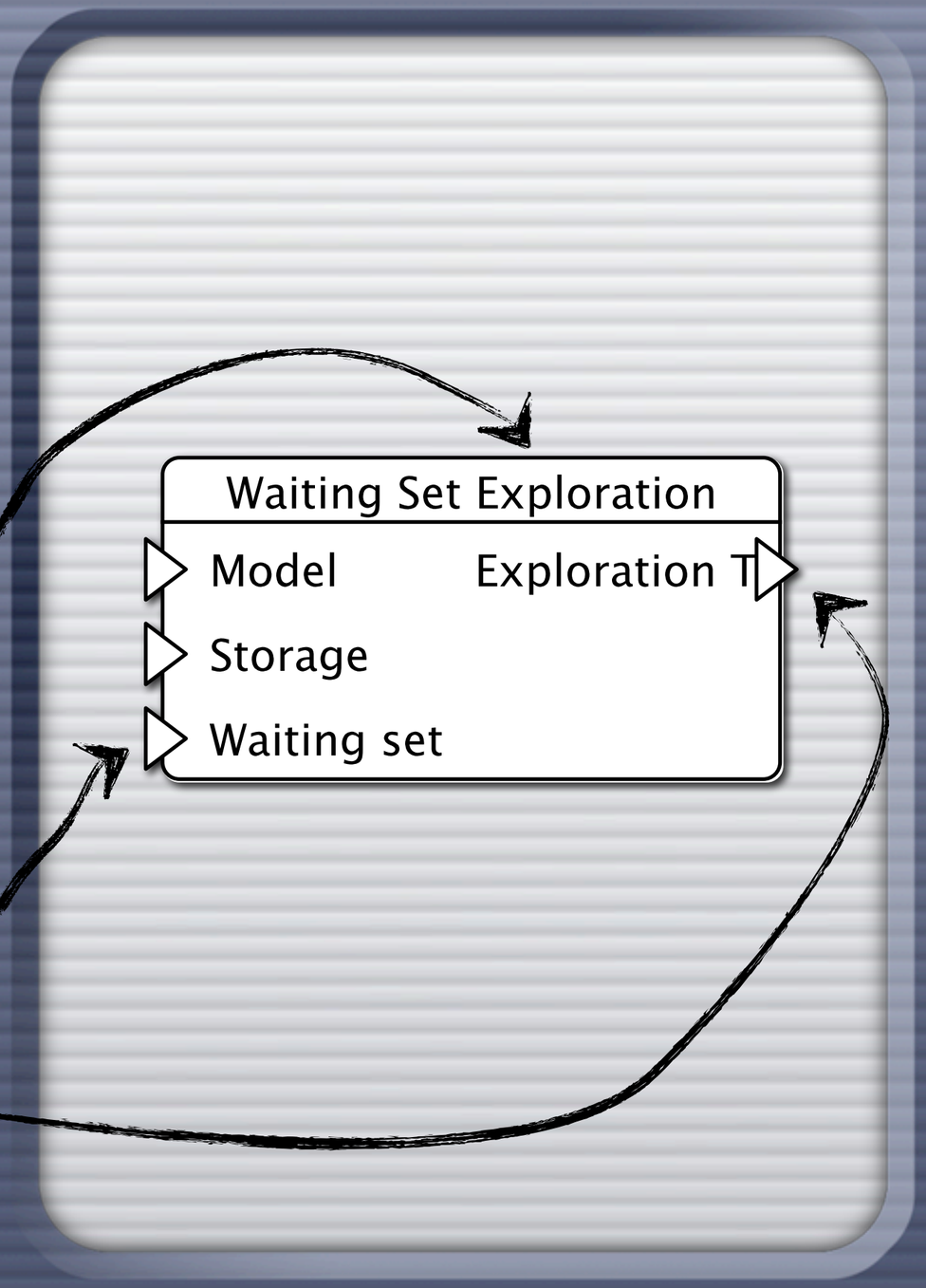
JoSEL Overview

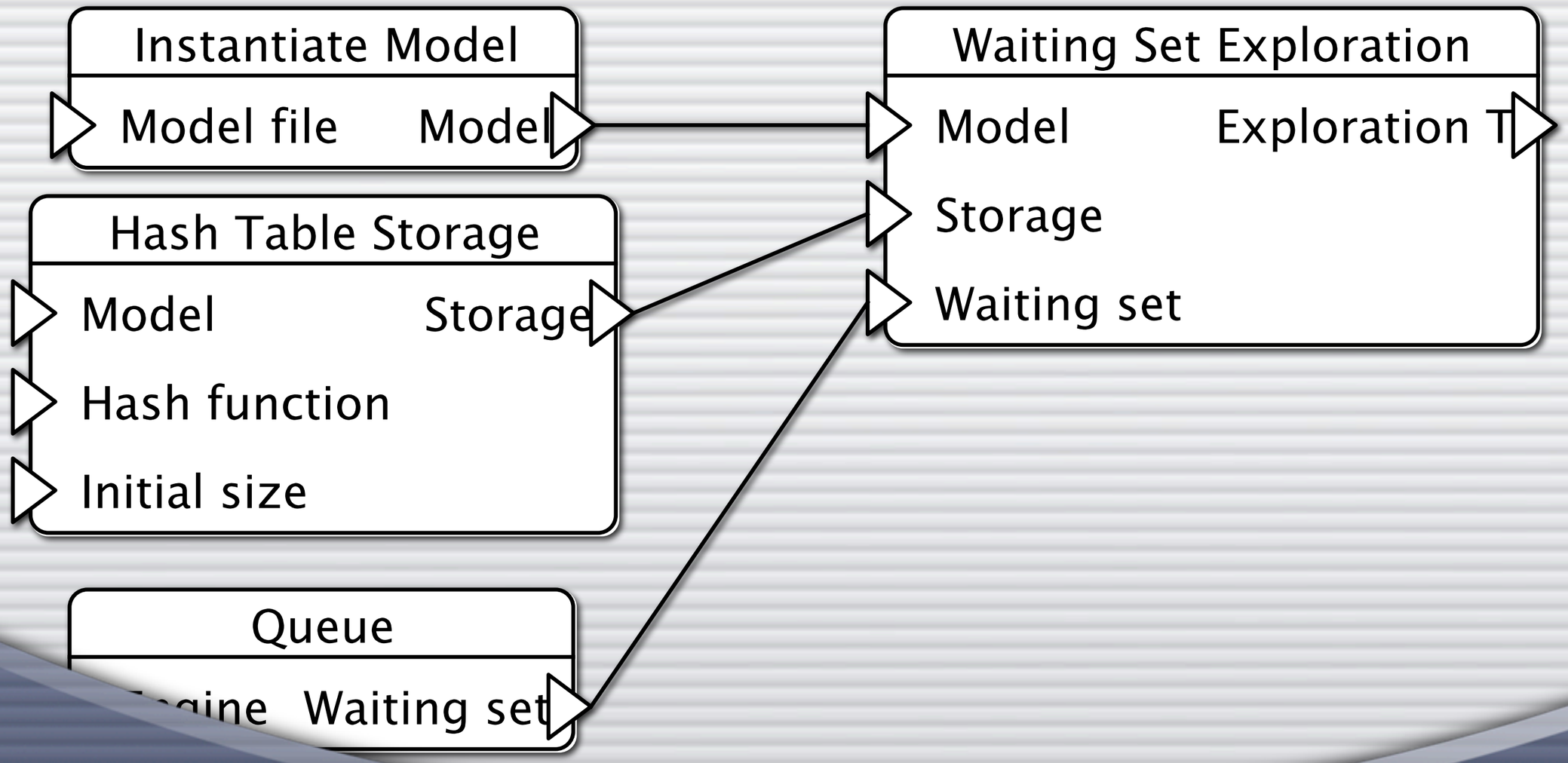
- Graphical language inspired by data-flow diagrams and hierarchy of CP-nets
- Basic unit is a **task**
- Tasks have typed input and output **ports**



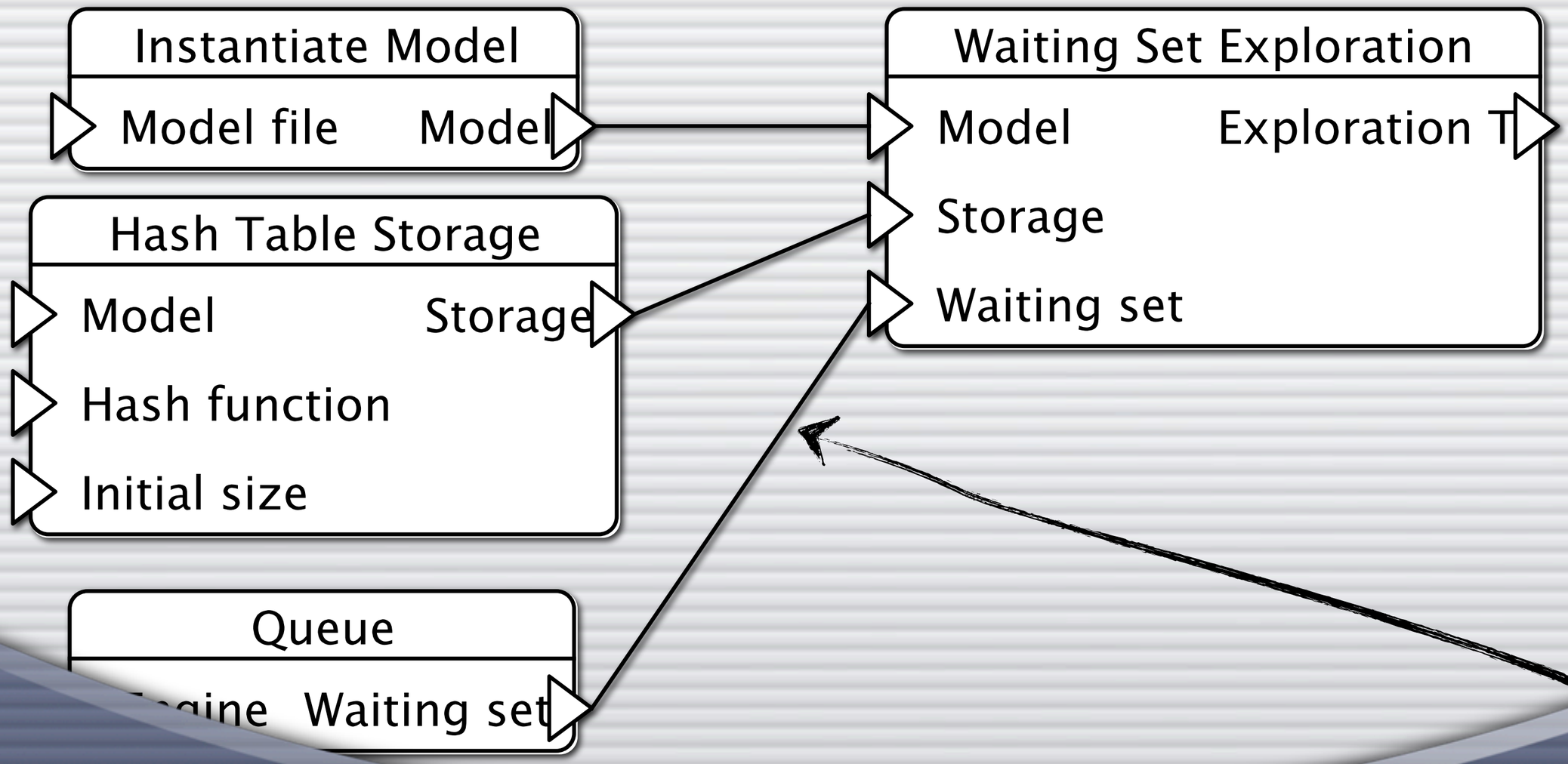
JoSEL Overview

- Graphical language inspired by data-flow diagrams and hierarchy of CP-nets
- Basic unit is a **task**
- Tasks have typed input and output **ports**

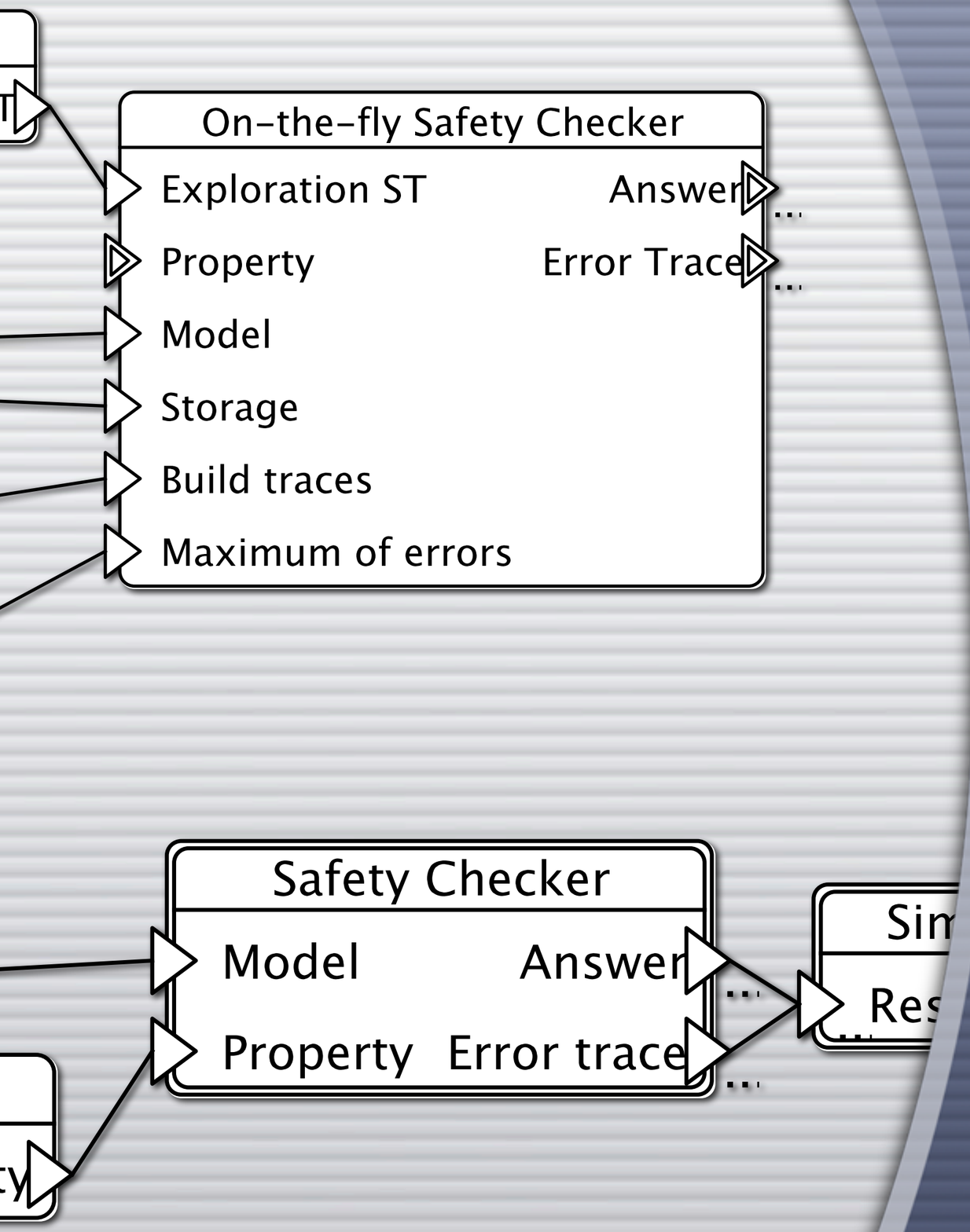




- Output and input ports can be **connected**
- A **verification job** (**job**) is a set of tasks and their connections

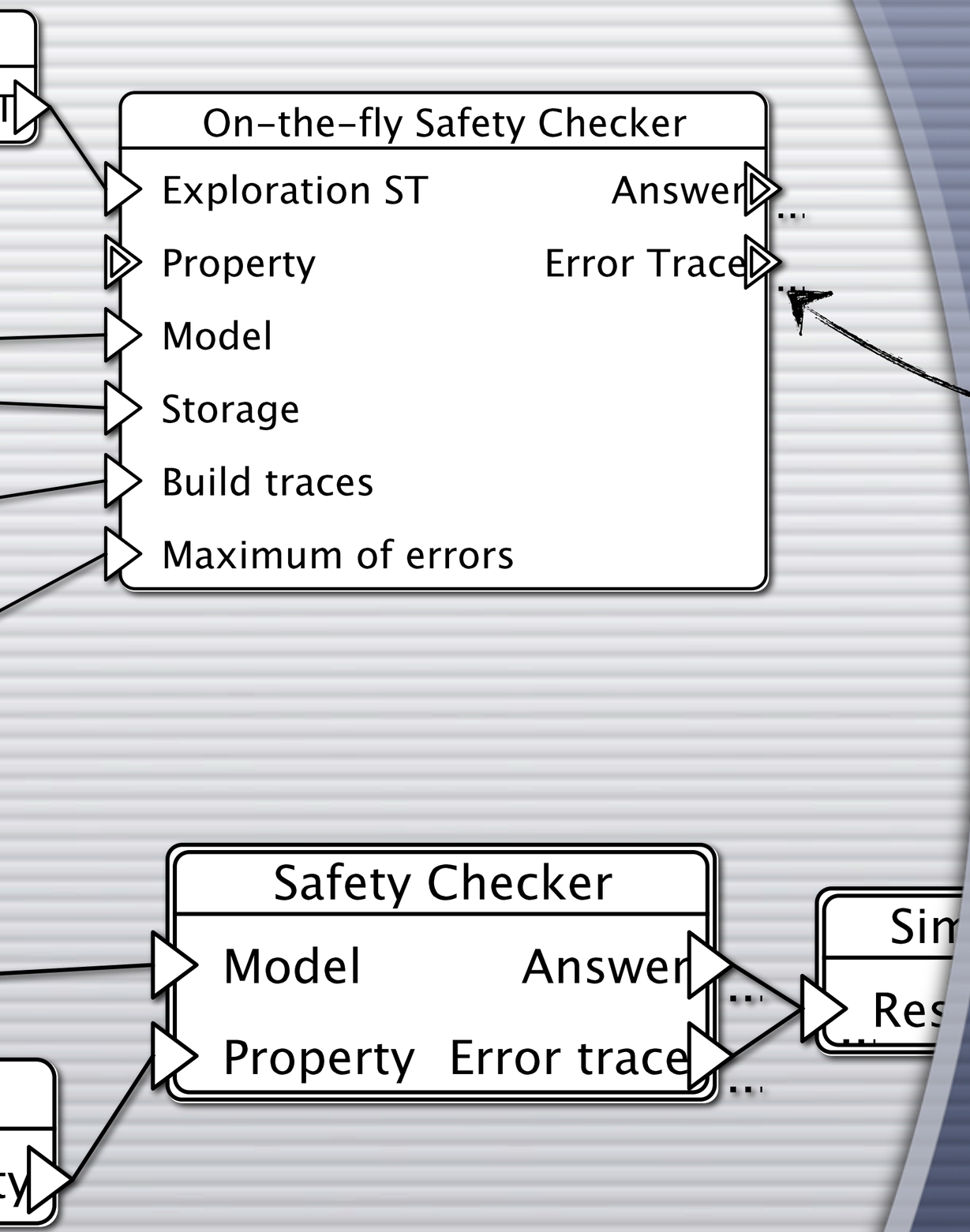


- Output and input ports can be **connected**
- A **verification job** (**job**) is a set of tasks and their connections



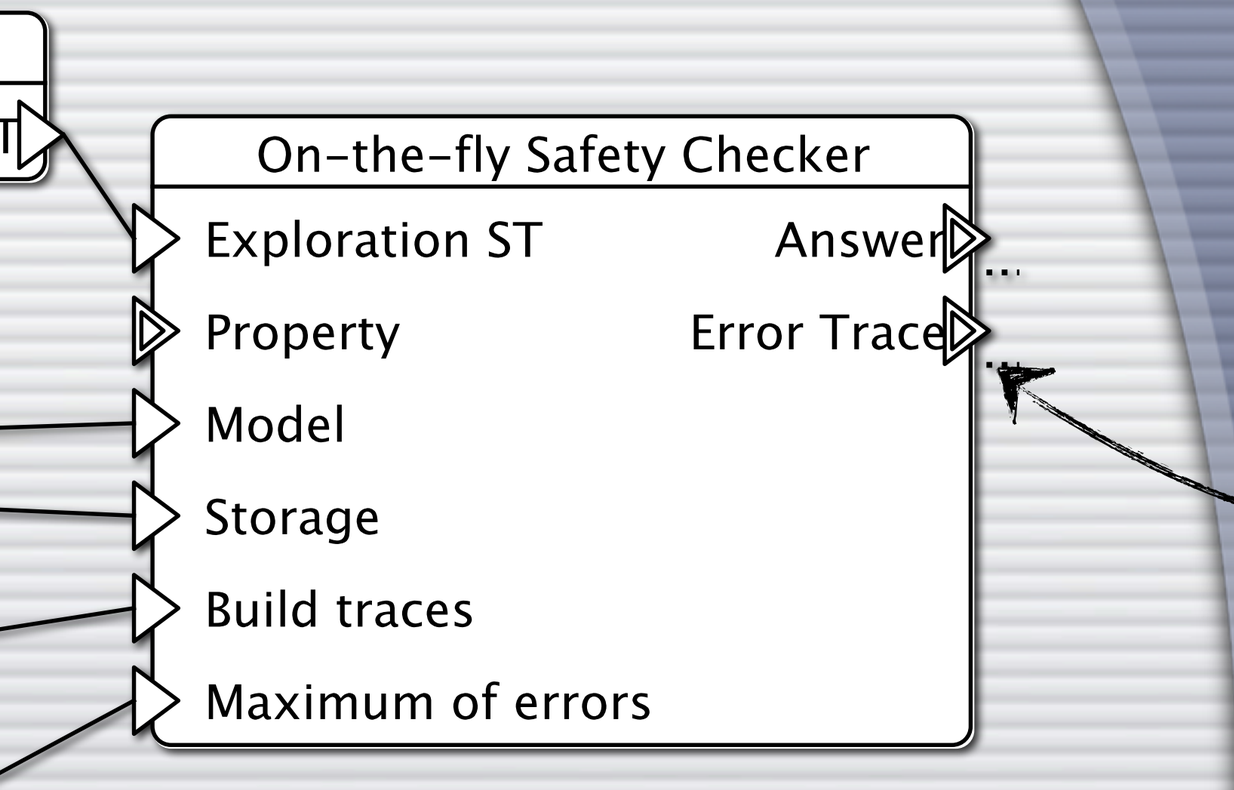
- Jobs can have **exported ports**

- Jobs can be represented by **macro tasks** (**macros**)

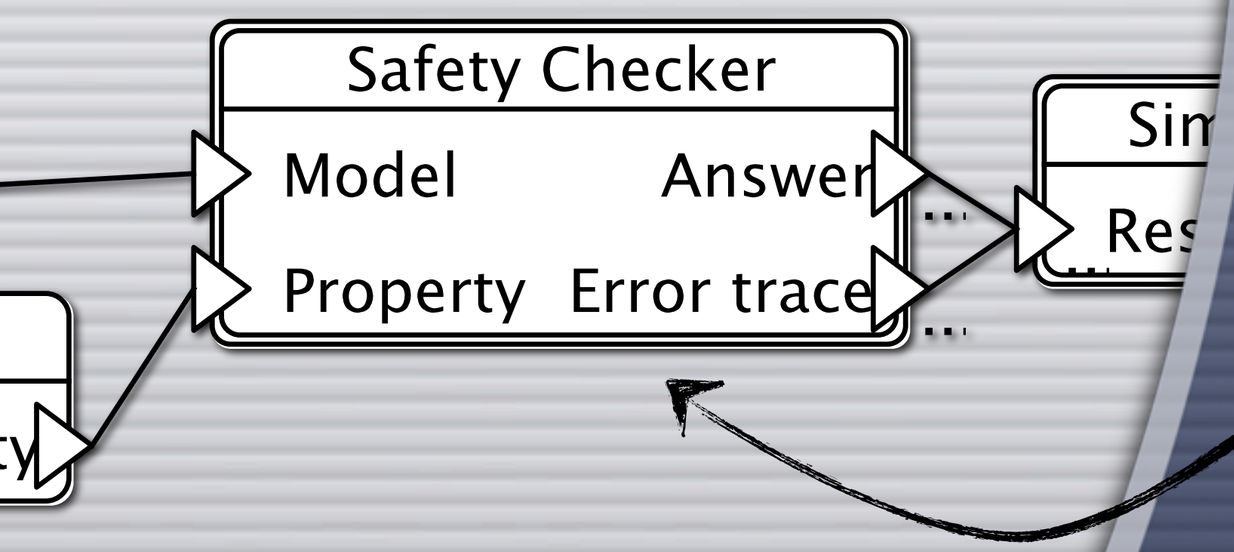


- Jobs can have **exported ports**

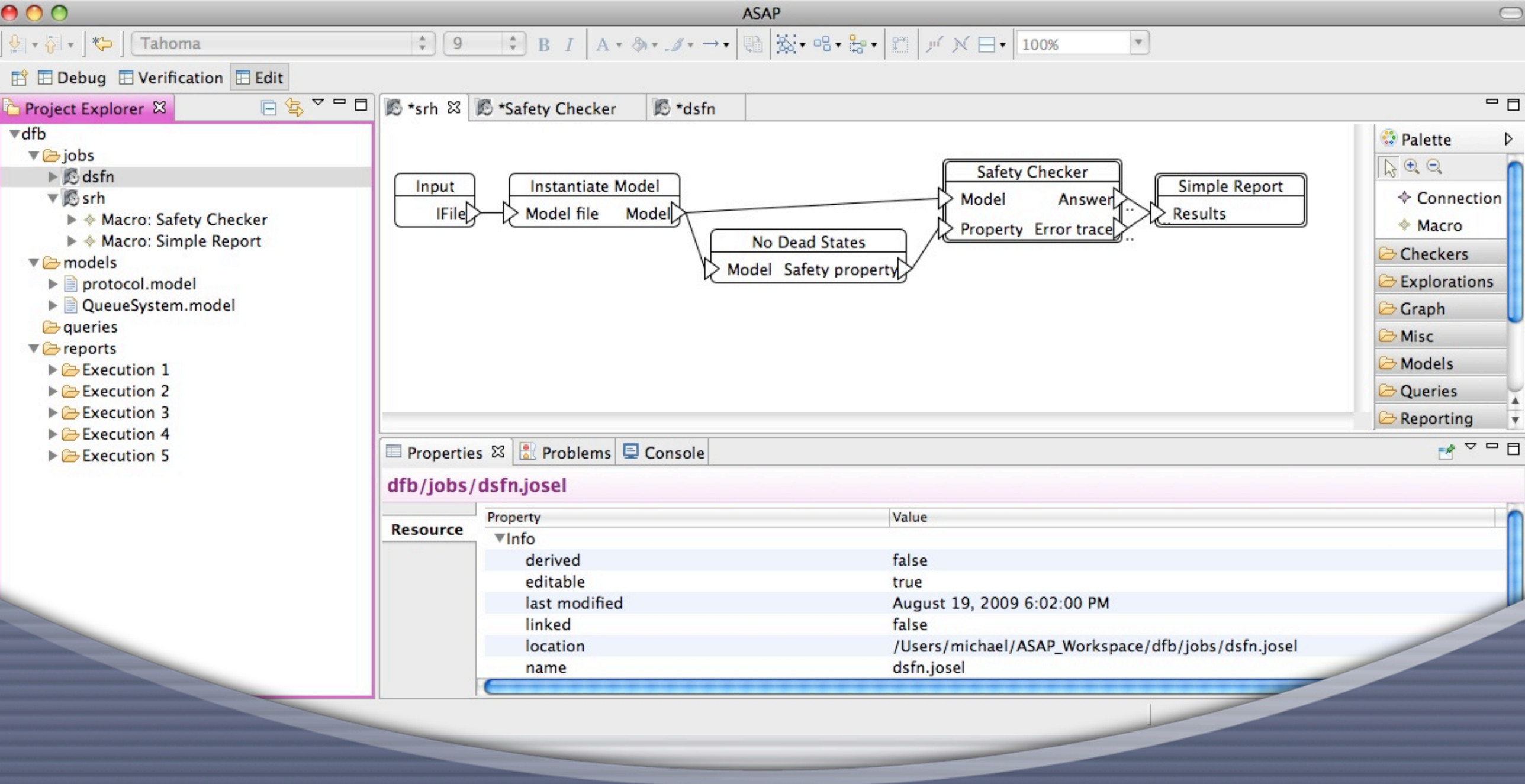
- Jobs can be represented by **macro tasks** (**macros**)



- Jobs can have **exported ports**

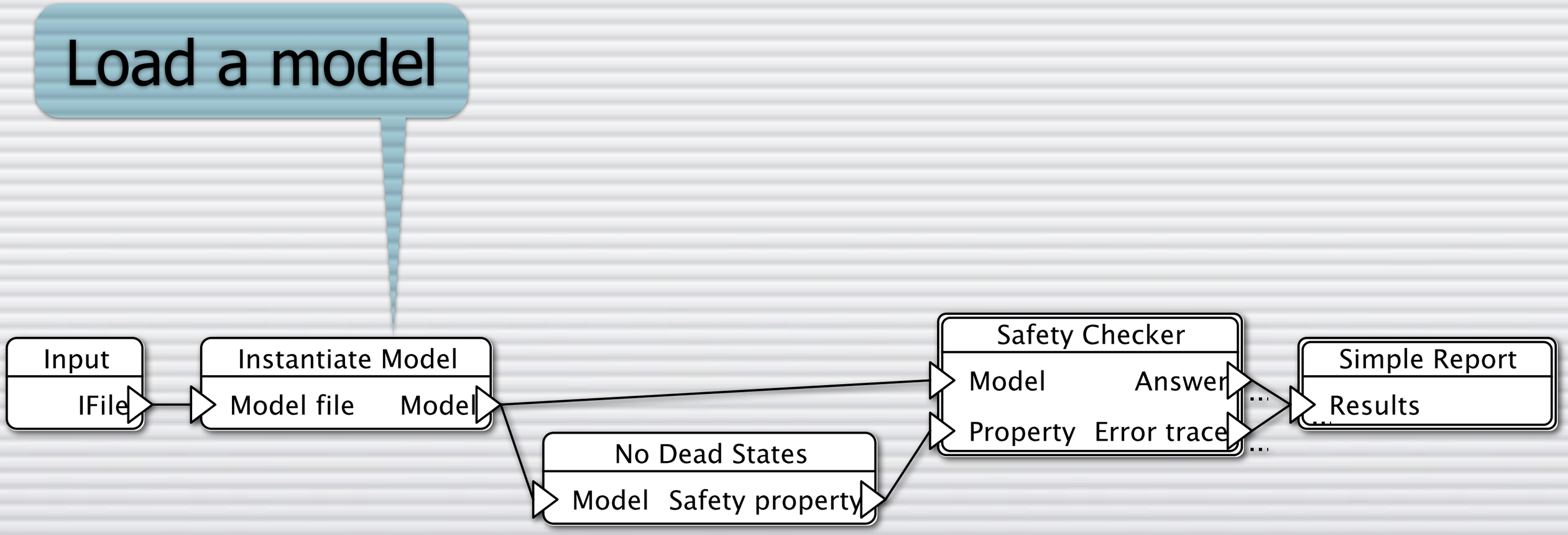


- Jobs can be represented by **macro tasks** (**macros**)

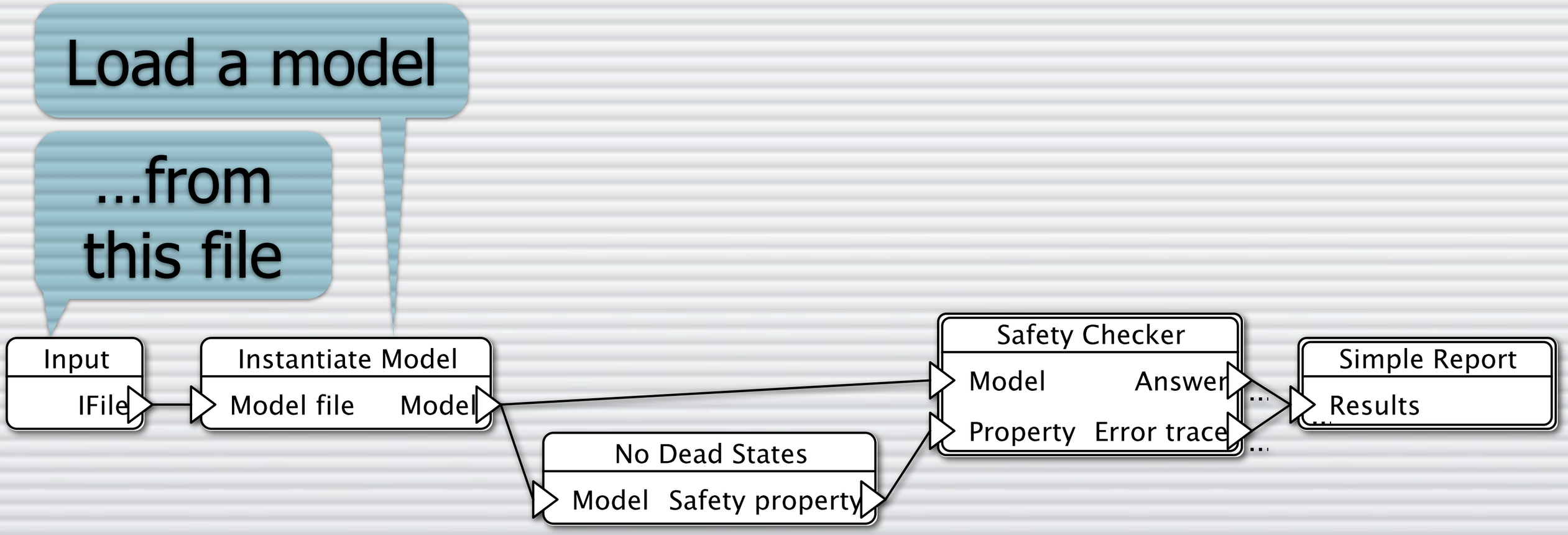


Deadlock Checker

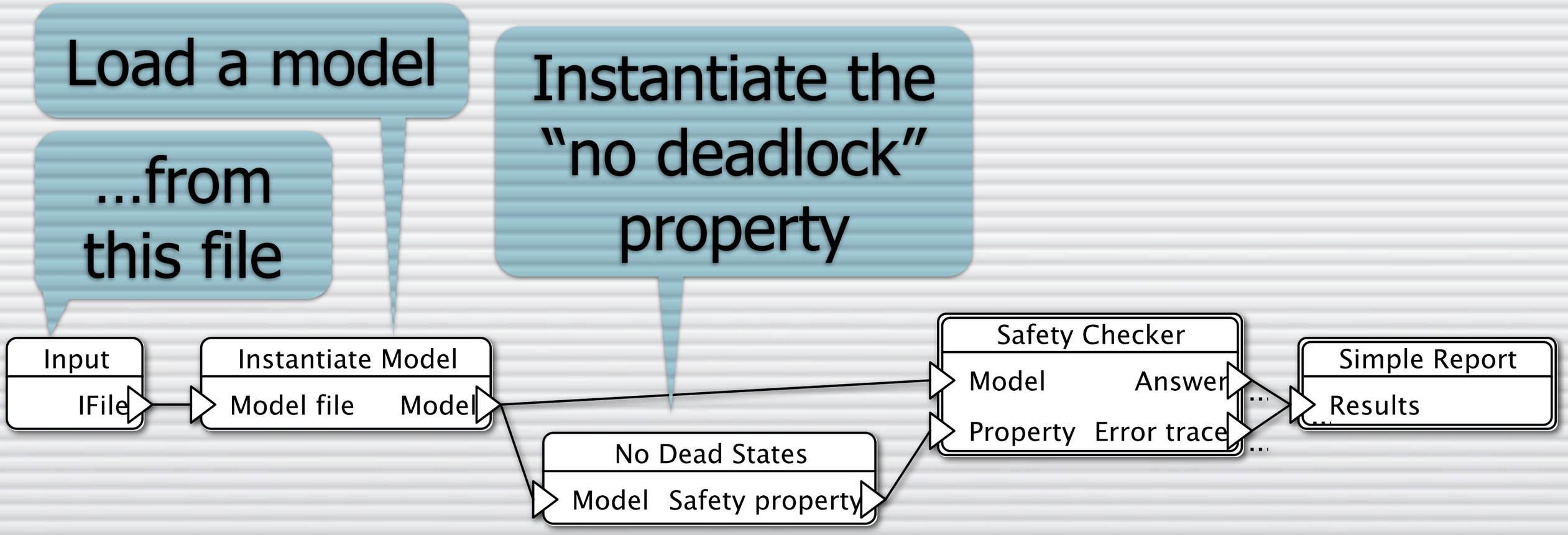
Deadlock Checker



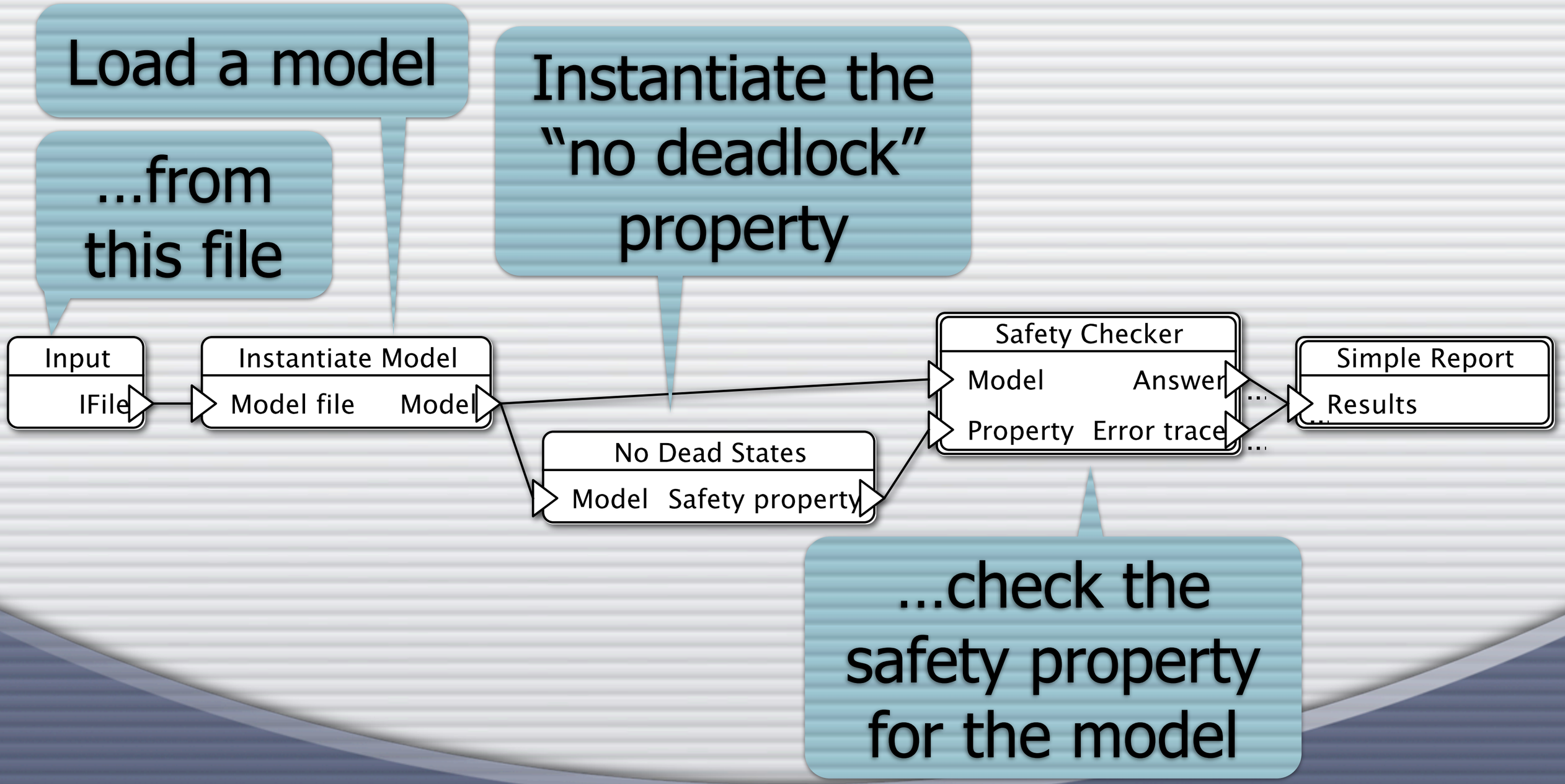
Deadlock Checker



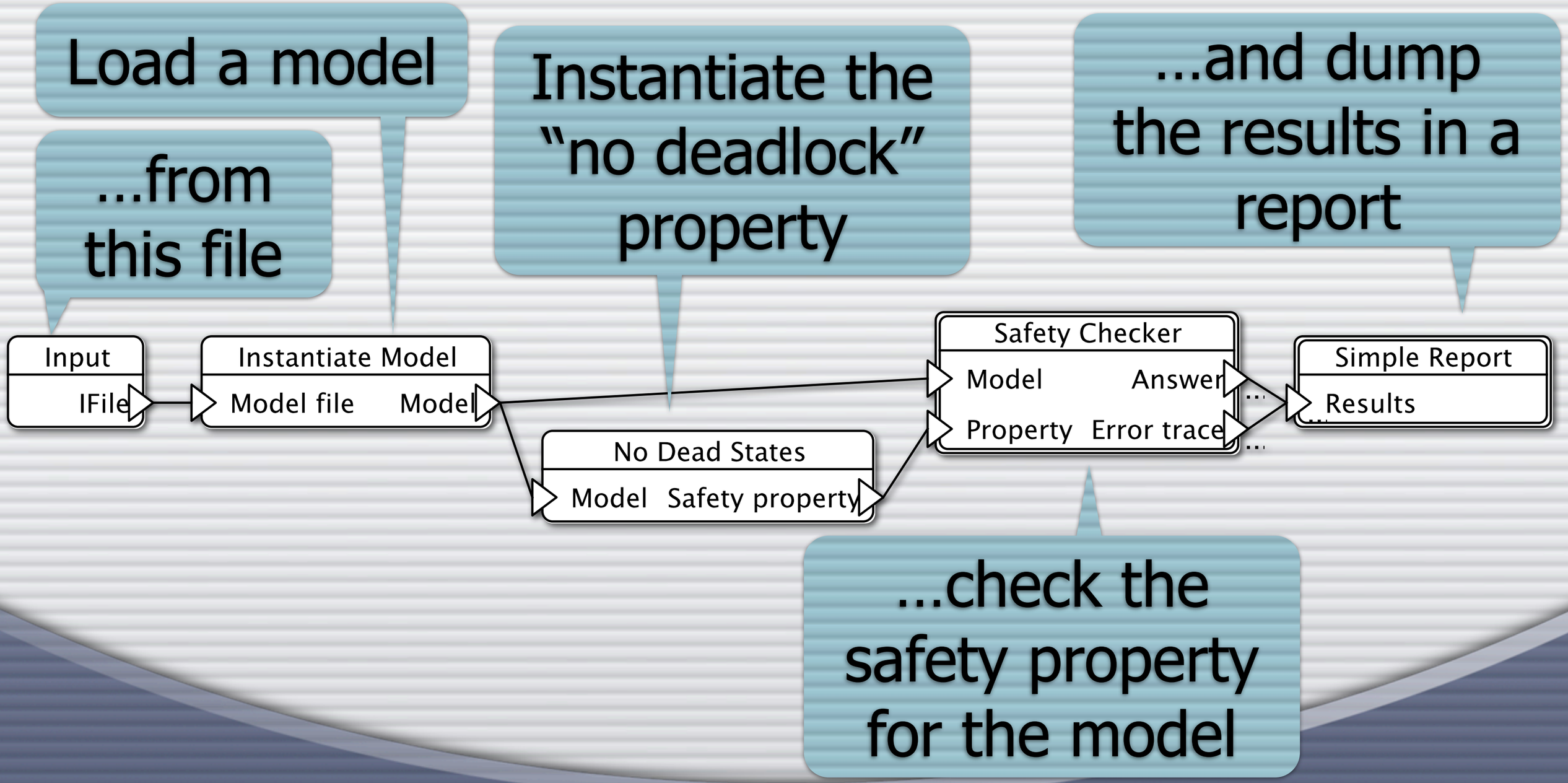
Deadlock Checker



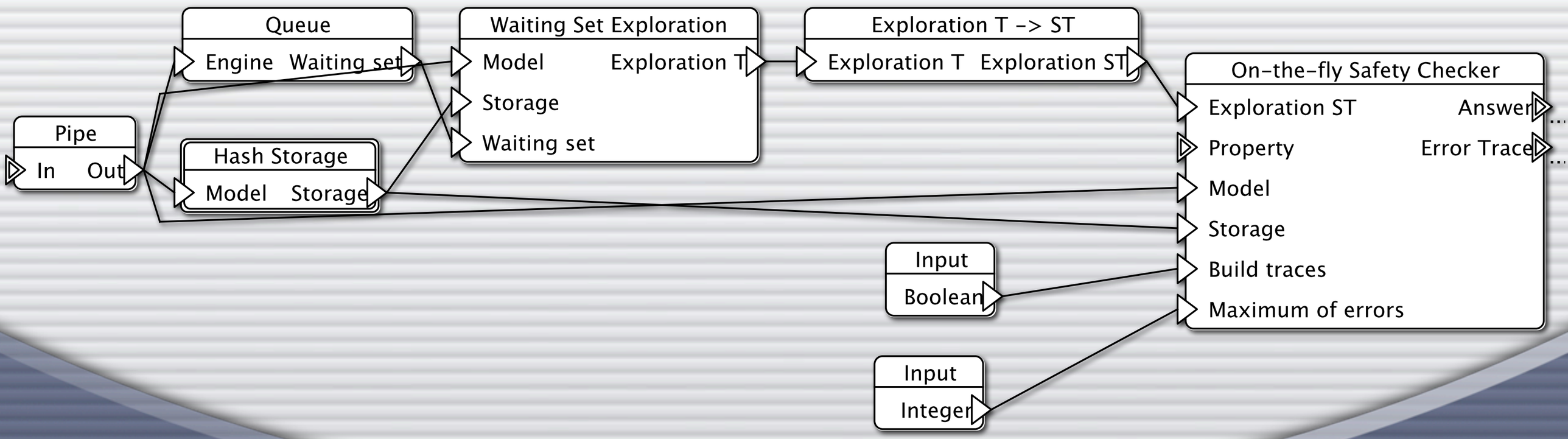
Deadlock Checker



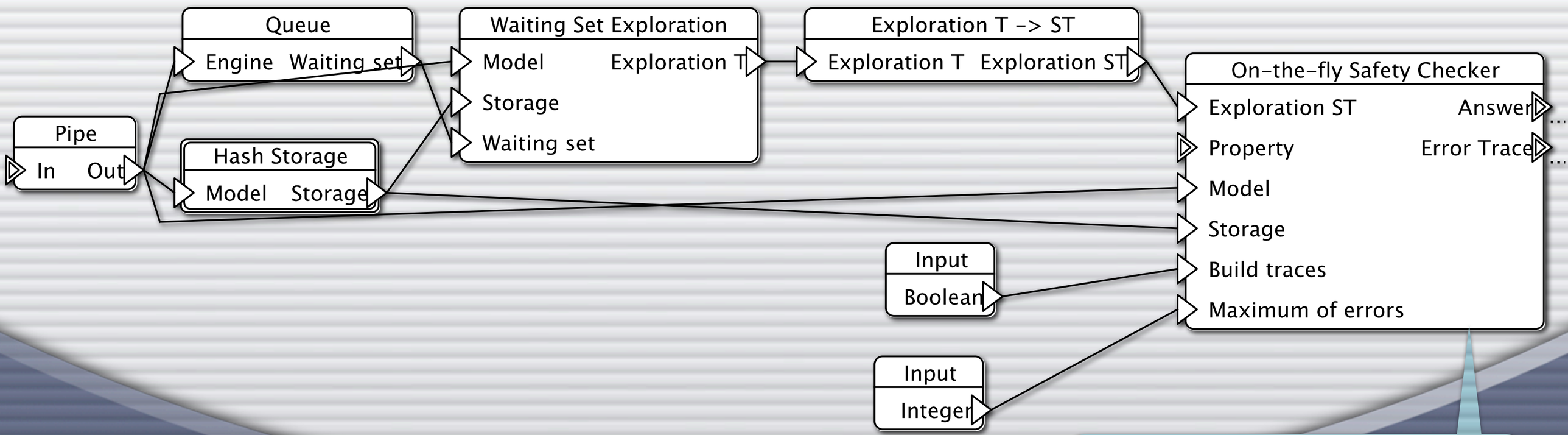
Deadlock Checker



Deadlock Checker

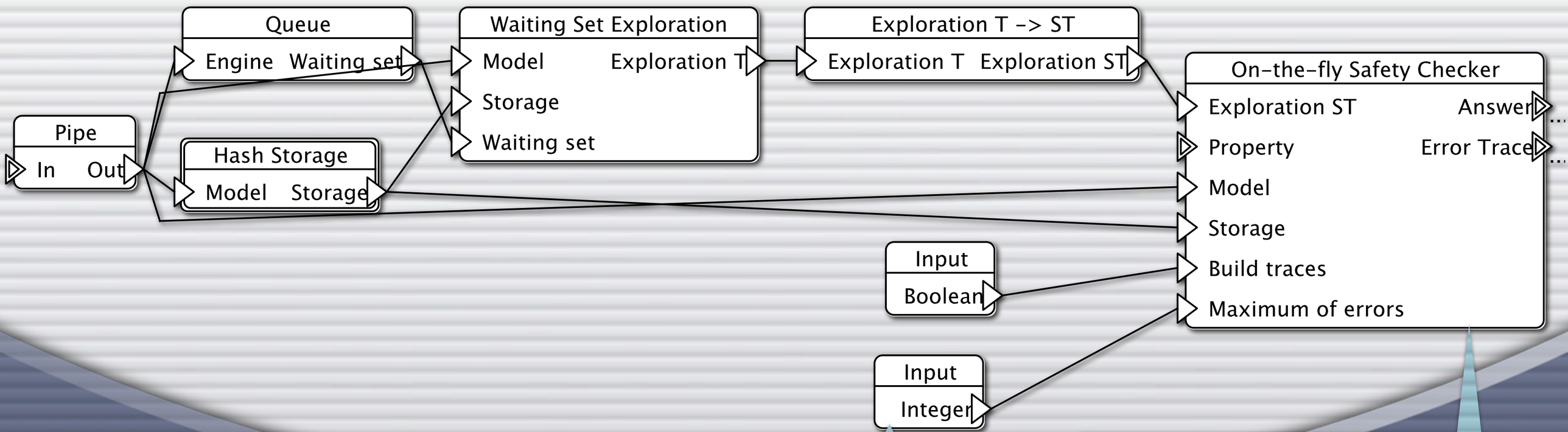


Safety Checker



Safety Check

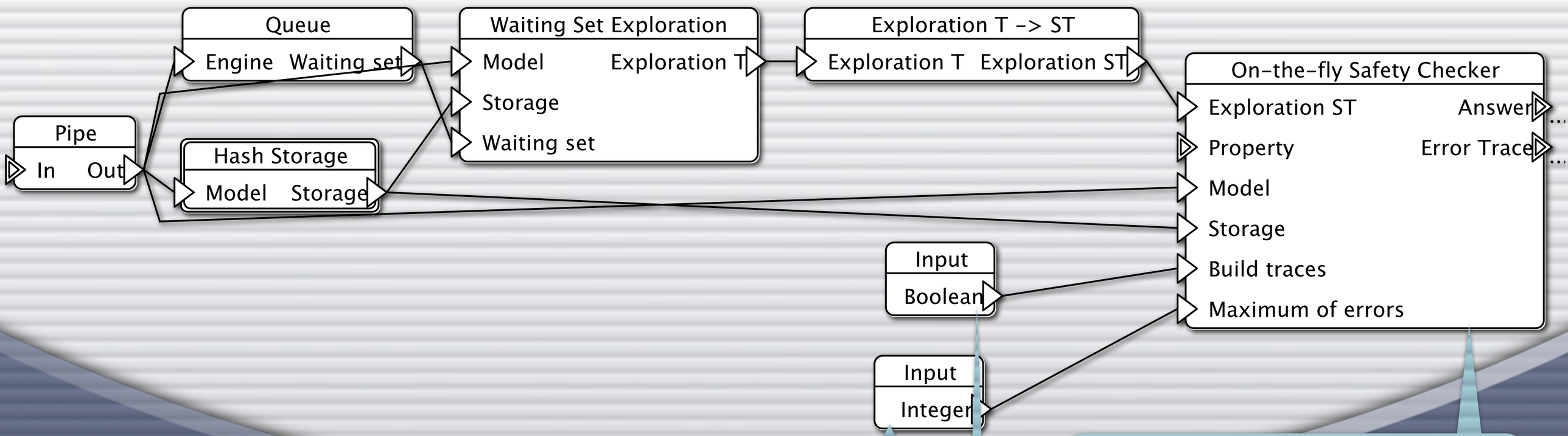
Check the given property using the given exploration



Stop after finding at most 10 errors

Check the given property using the given exploration

Safety Check

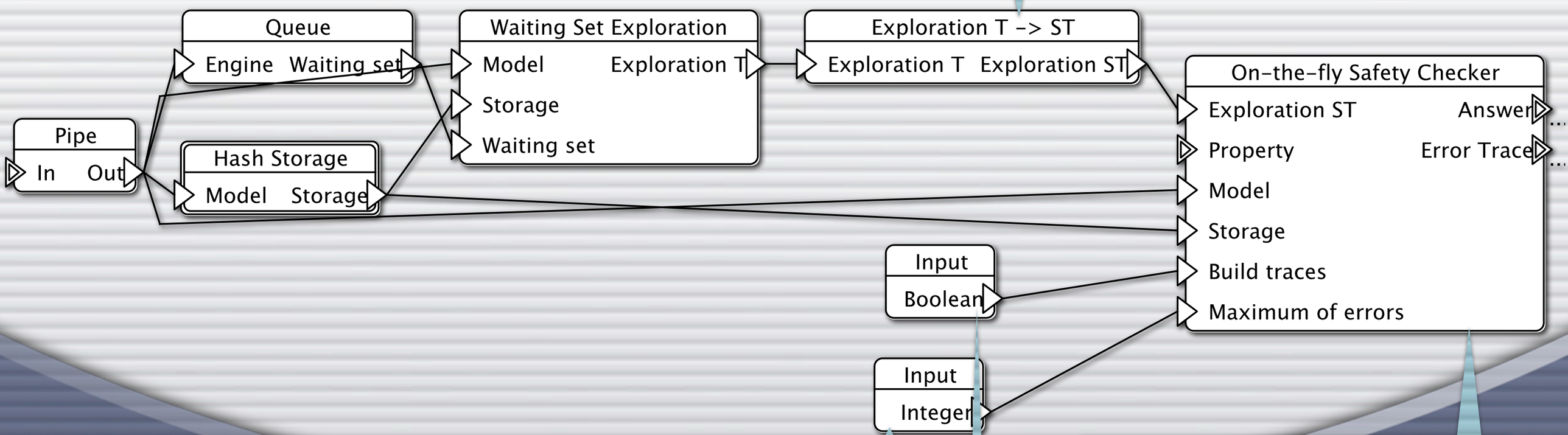


Stop after finding at most 10 errors

Check the given property using the given exploration

Build error-traces during exploration

Technical – just make sure the letters match



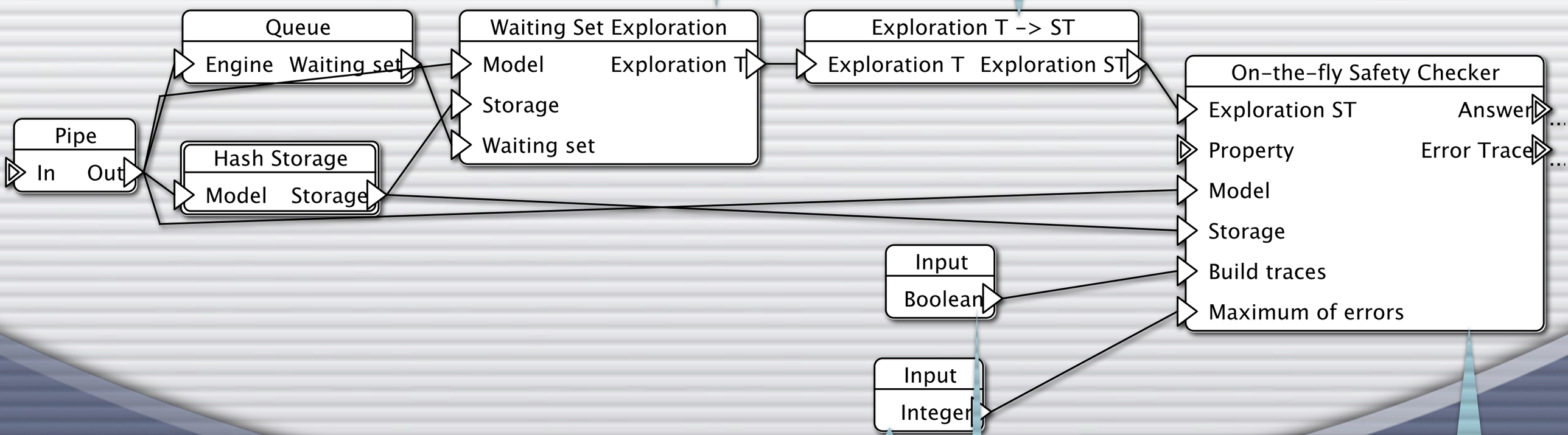
Stop after finding at most 10 errors

Check the given property using the given exploration

Build error-traces during exploration

Exploration algorithm

Technical – just make sure the letters match



Stop after finding at most 10 errors

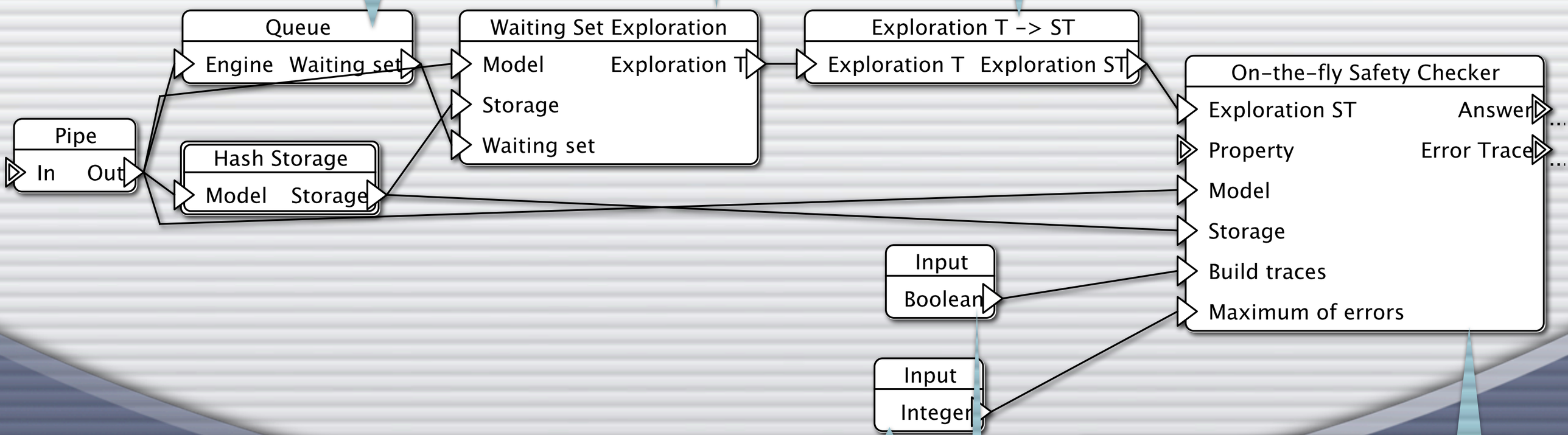
Check the given property using the given exploration

Build error-traces during exploration

Temporary storage is a queue

Exploration algorithm

Technical – just make sure the letters match



Stop after finding at most 10 errors

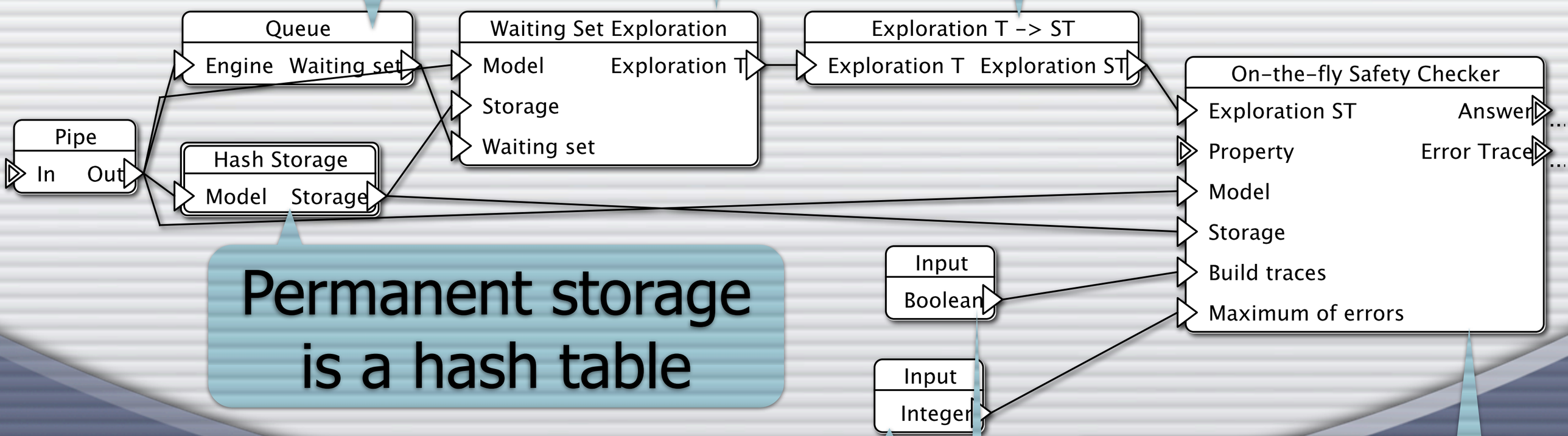
Check the given property using the given exploration

Build error-traces during exploration

Temporary storage is a queue

Exploration algorithm

Technical – just make sure the letters match



Permanent storage is a hash table

Stop after finding at most 10 errors

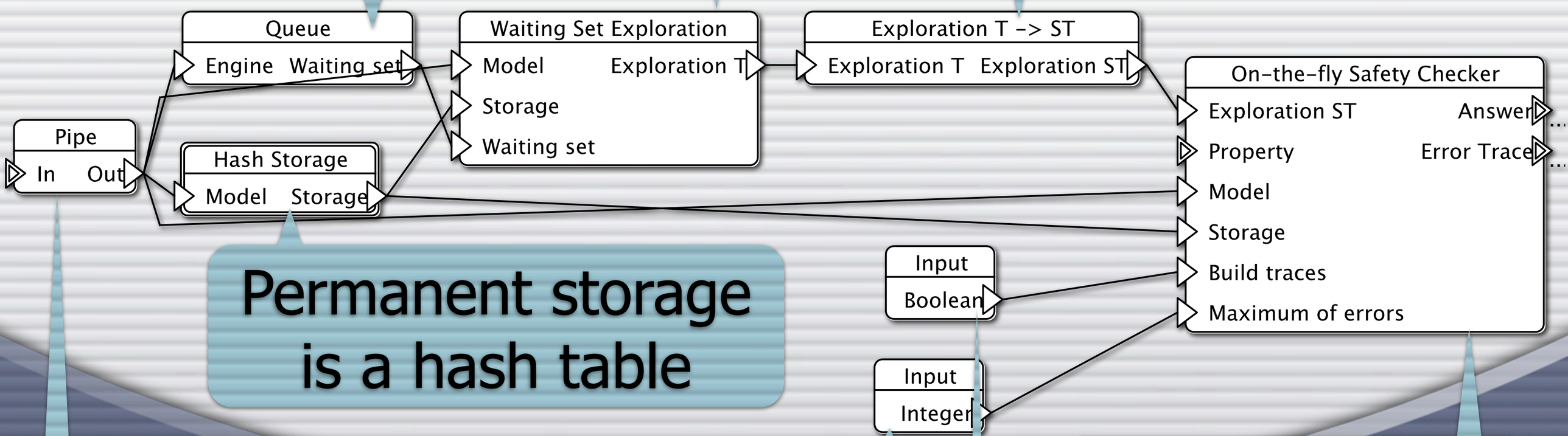
Check the given property using the given exploration

Build error-traces during exploration

Temporary storage is a queue

Exploration algorithm

Technical – just make sure the letters match



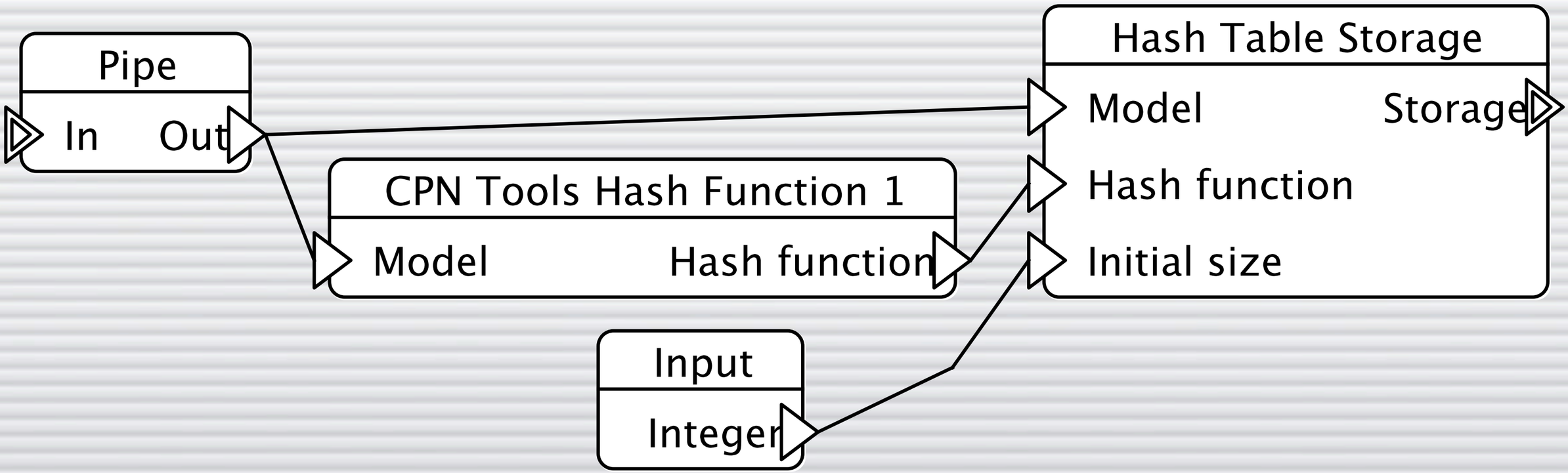
Permanent storage is a hash table

Technical – allows us to only specify the model once on the level above

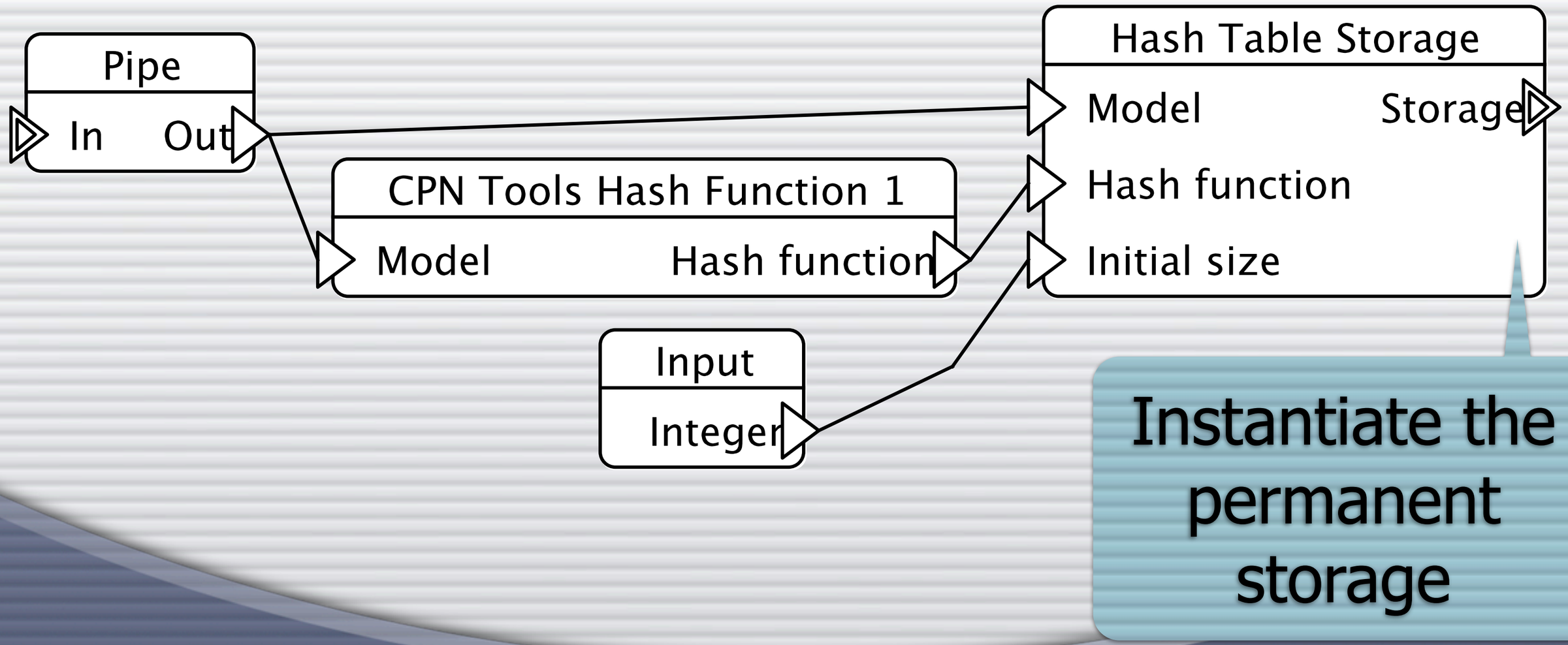
Stop after finding at most 10 errors

Build error-traces during exploration

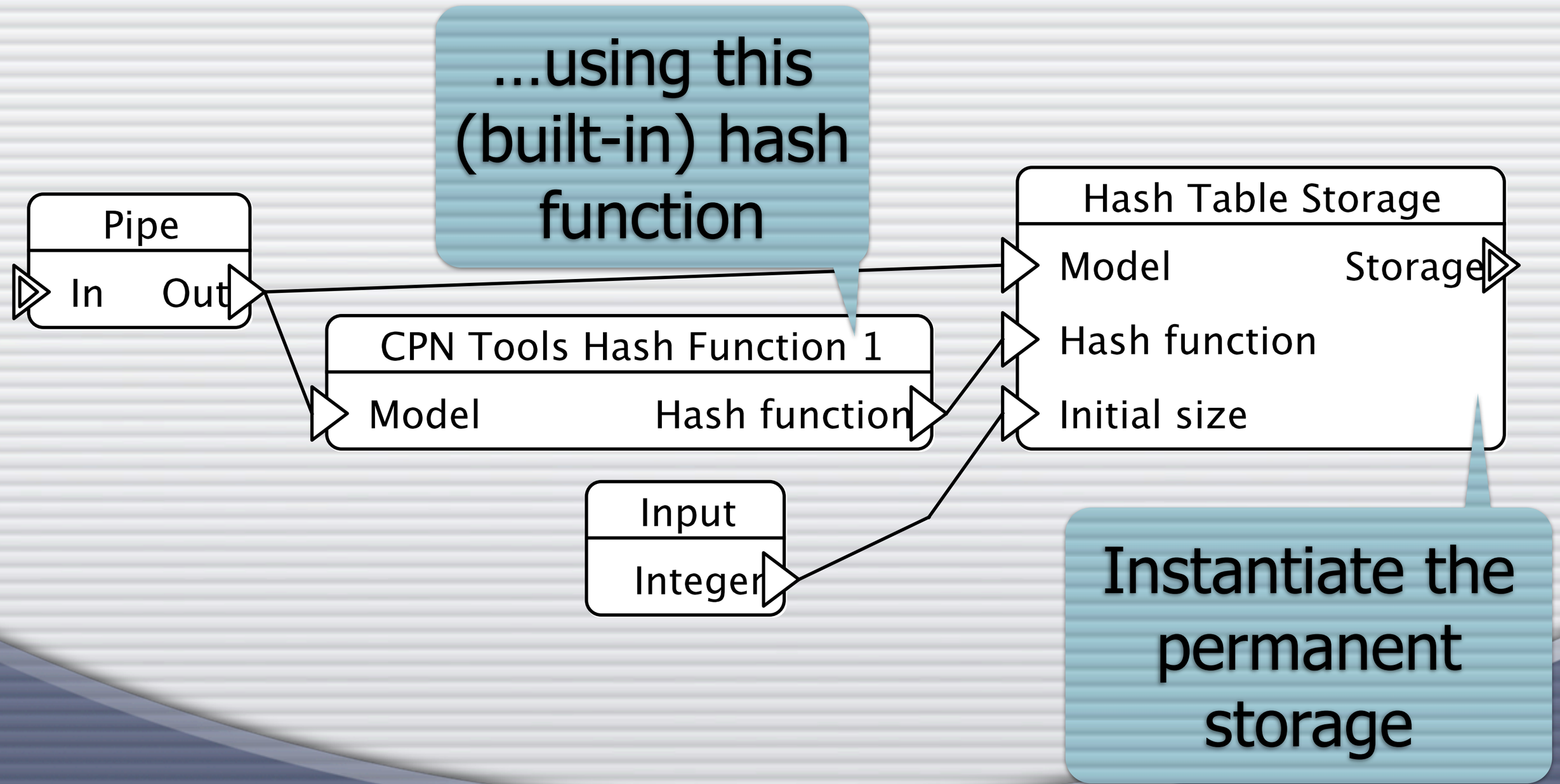
Check the given property using the given exploration



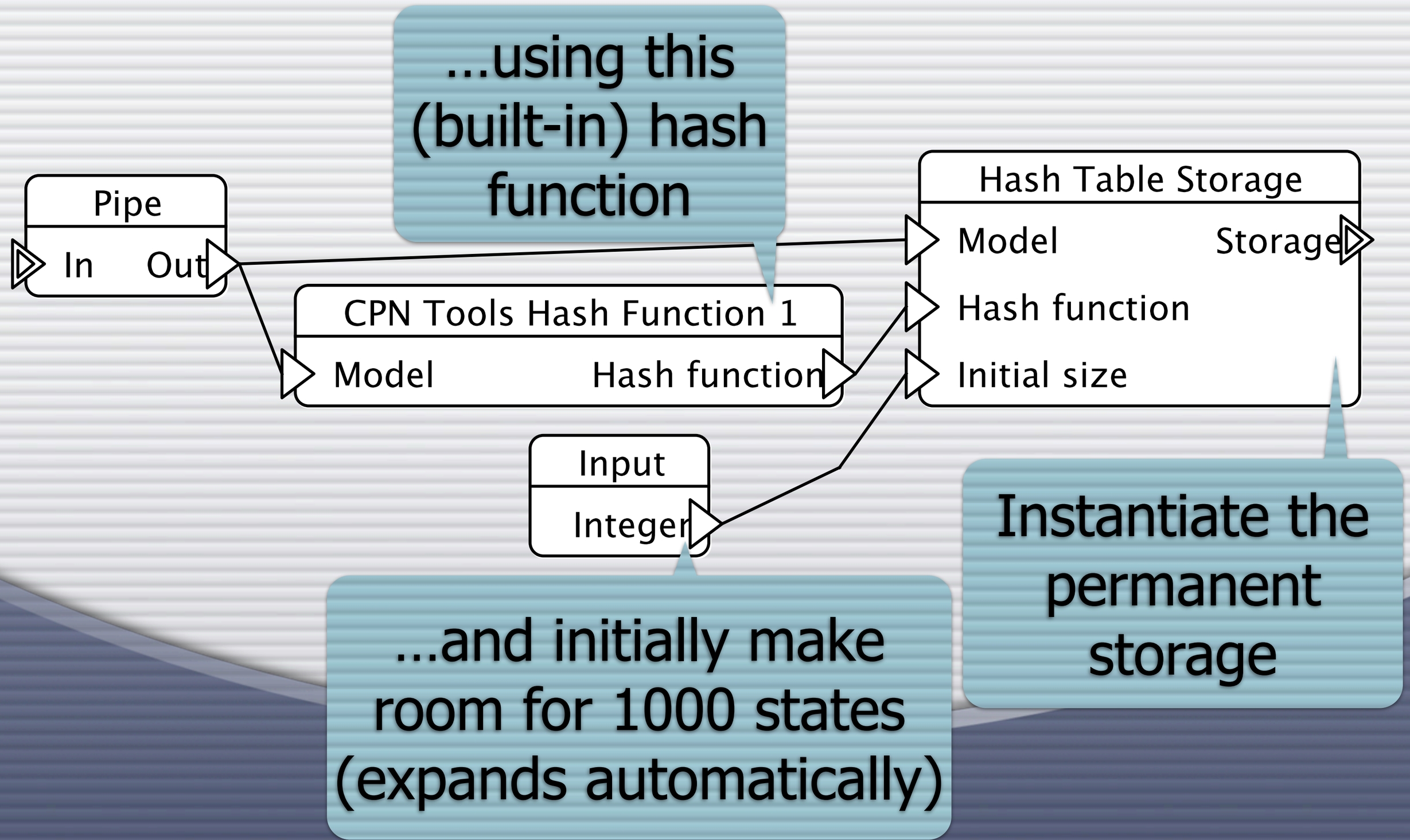
Hash Storage



Hash Storage

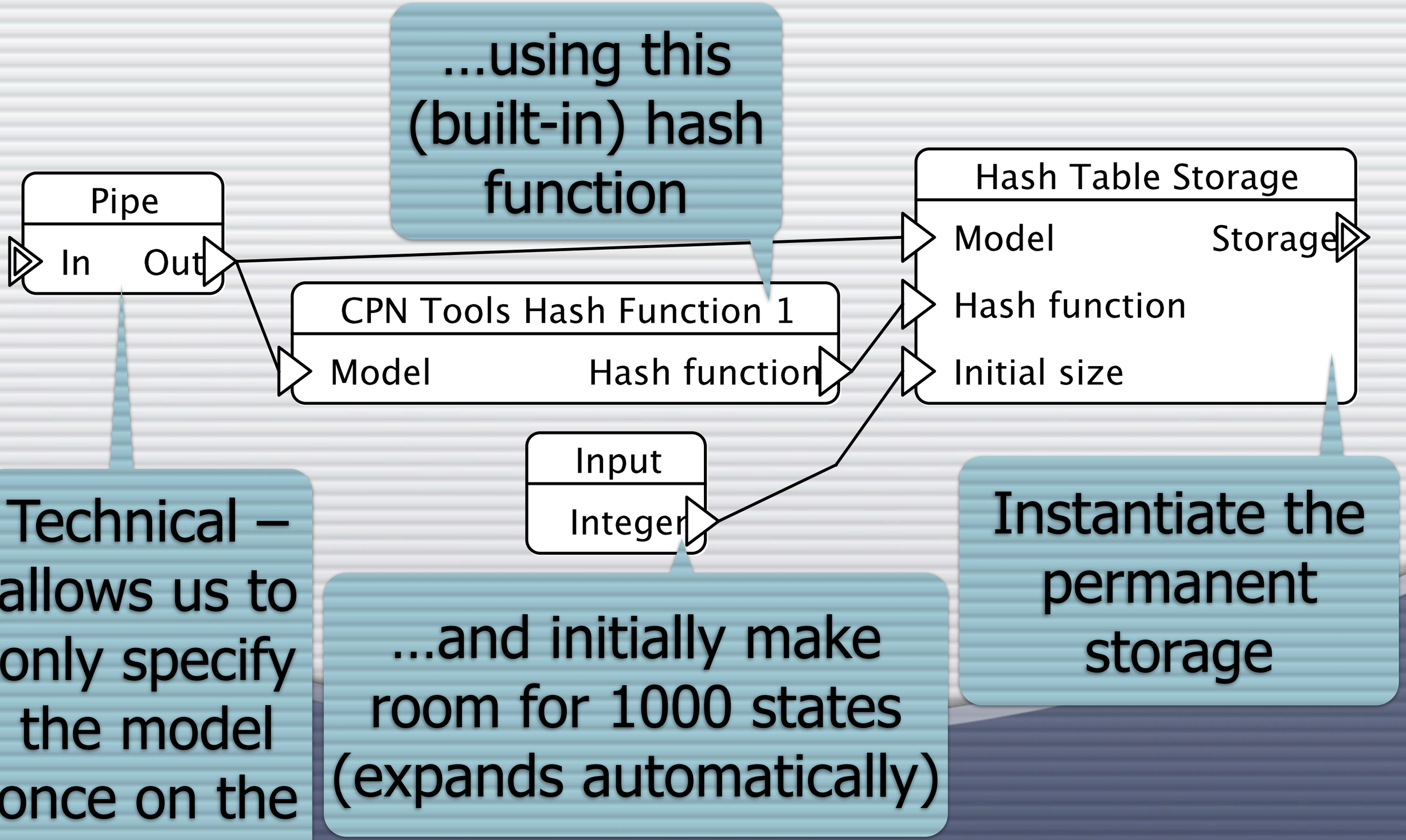


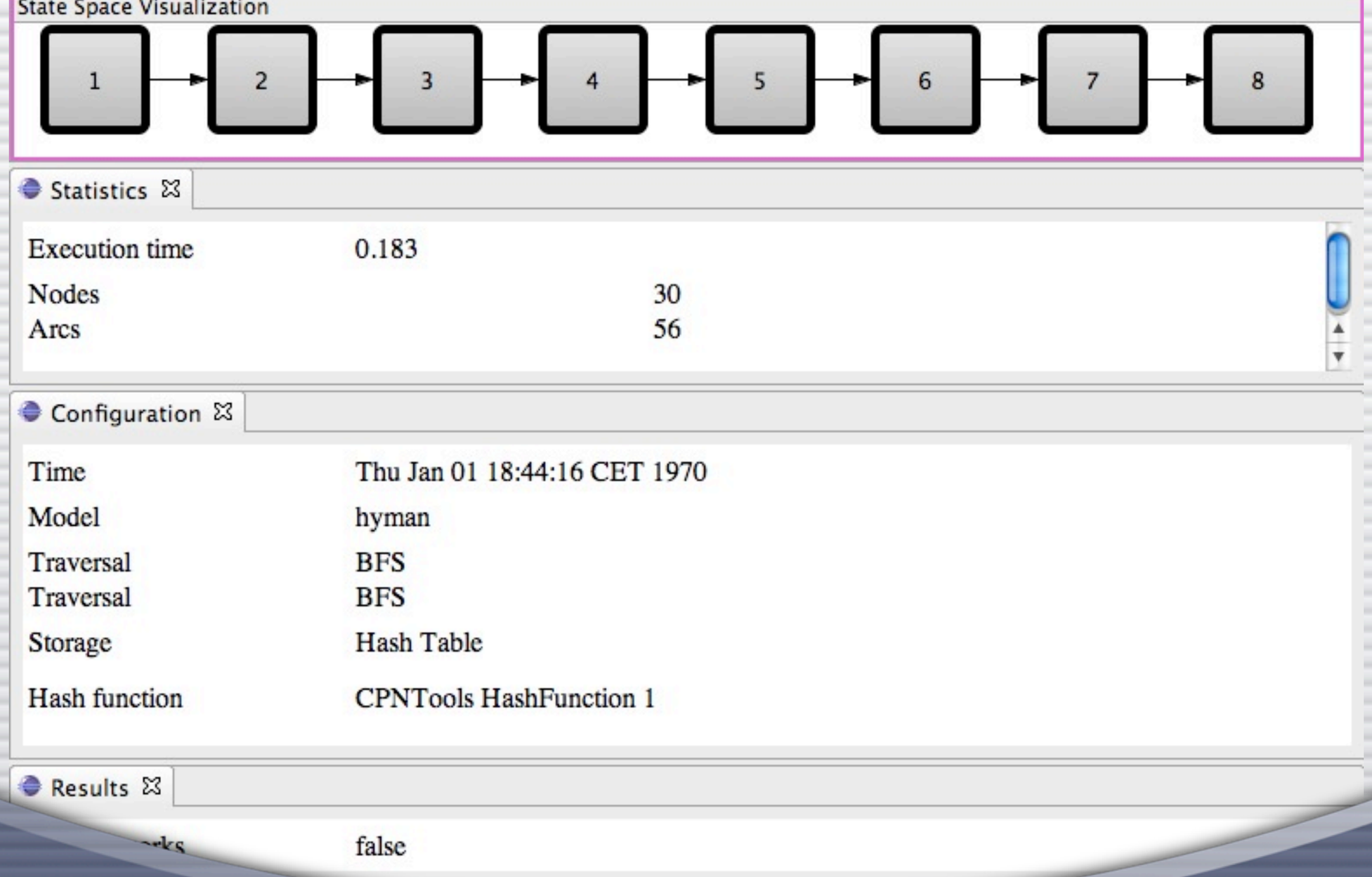
Hash Storage



Hash Storage

Hash Storage





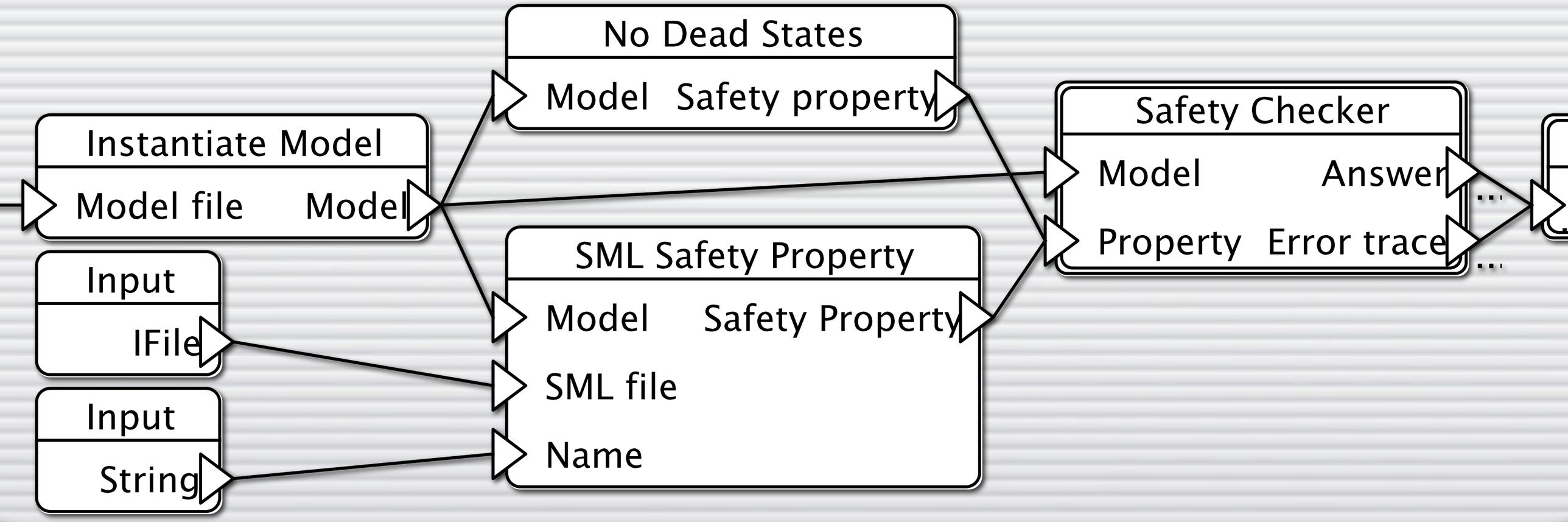
Example: Playing with JoSEL

Demo:
Playing with JoSEL (03)

- Displaying error trace
  - Displaying multiple error traces in a single window

Custom Properties

- Some times we may want to check properties other than absence of deadlocks
- Custom properties are created using SML
- ASAP automatically generates a template formula tailored to a specific model



Example: Mutual Exclusion

Example: Mutual Exclusion

- We want to check that two adjacent philosophers cannot be eating at the same time
  - I.e., that they are not allowed access to a shared resource (chop-stick) at the same time
- This is equivalent to checking that if philosopher p is eating, then philosopher p+1 is not (mod n)

A Bit of SML

- Check if there is an element "p" in "lst" that satisfies the predicate "f(p)":
```
List.exists (fn p => f(p)) lst
```
- Check if "2 + 1 mod 7" belongs to a list, "lst":
```
List.exists (fn p' => p' = (2 + 1) mod 7) lst
```
- Check if "p + 1 mod n" belongs to a list, "lst":
```
List.exists (fn p' => p' = (p + 1) mod n) lst
```
- Check if there is an element "p" in "lst" such that "p + 1 mod n" belongs to "lst":
```
List.exists (fn p => List.exists
  (fn p' => p' = (p + 1) mod n) lst) lst
```

Yes, this is inefficient; we can sort "lst" and only compare neighbors

Example: Mutual Exclusion

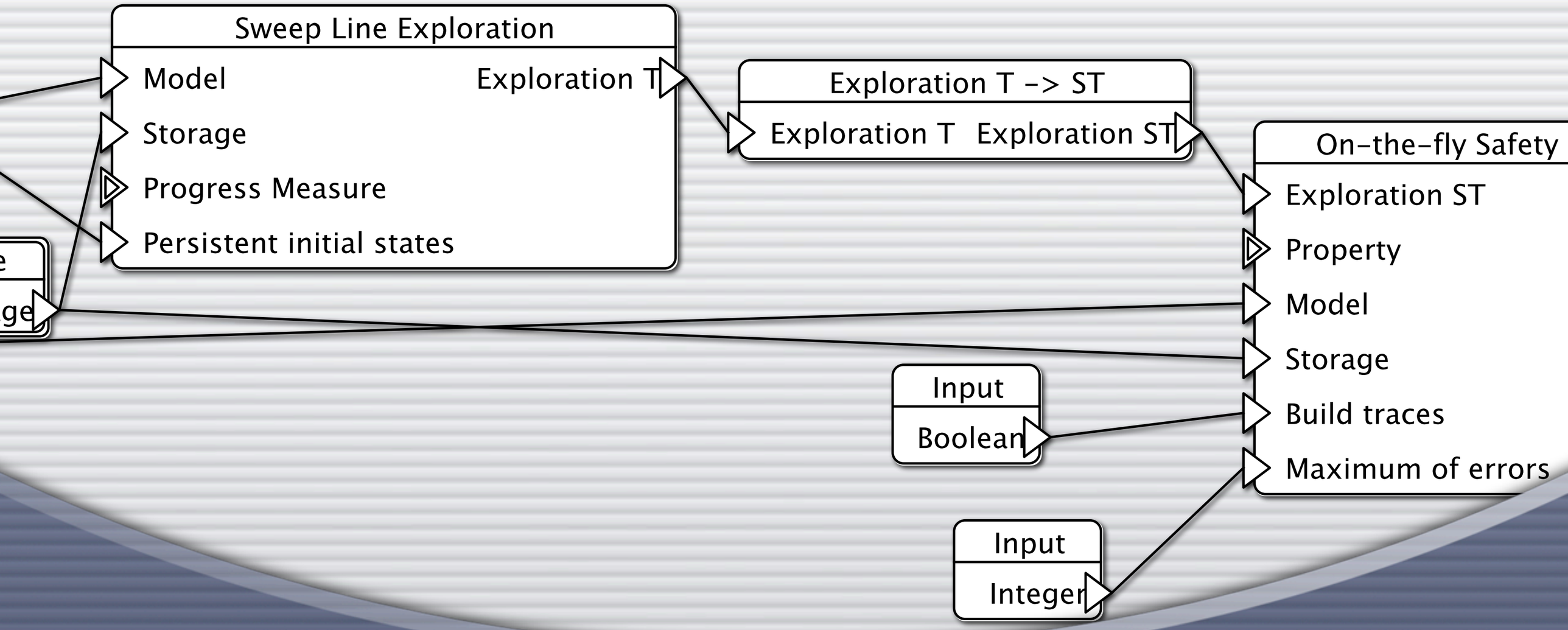
```
fun query (state, events) =
  let
    fun query'New_Page { Waiting, Has_One, Eating,
                         Philosophers, Initialized,
                         Chopsticks } = true
    fun query'state { New_Page} = query'New_Page New_Page
  in
    query'state state
  end
```

Example: Mutual Exclusion

```
fun query (state, events) =
let
  fun query'New_Page { Waiting, Has_One, Eating,
                       Philosophers, Initialized,
                       Chopsticks } =
    not (List.exists (fn p => List.exists
                  (fn p' => p’ =
                    (p + 1) mod (List.hd Philosophers)
                  ) Eating) Eating)
  fun query'state { New_Page} = query'New_Page New_Page
in
  query'state state
```

Demo: Mutual Exclusion (04)

- Create property
- Edit JoSEL job
- Run checker



Example: The Sweep-line Method

State Space Methods

- Store states compactly
- Delete states during exploration
- Store only some states
- Use external memory

State Space Methods

- Store states compactly
- Delete states during exploration
- Store only some states
- Use external memory

Briefly: The Sweep-line Method

- Uses notion of progress in model identified by a **progress measure**
- A conceptual **sweep-line** marks a border between states that have already been discovered
- Only states in front of the sweep-line is kept in memory

Briefly: A Progress Measure

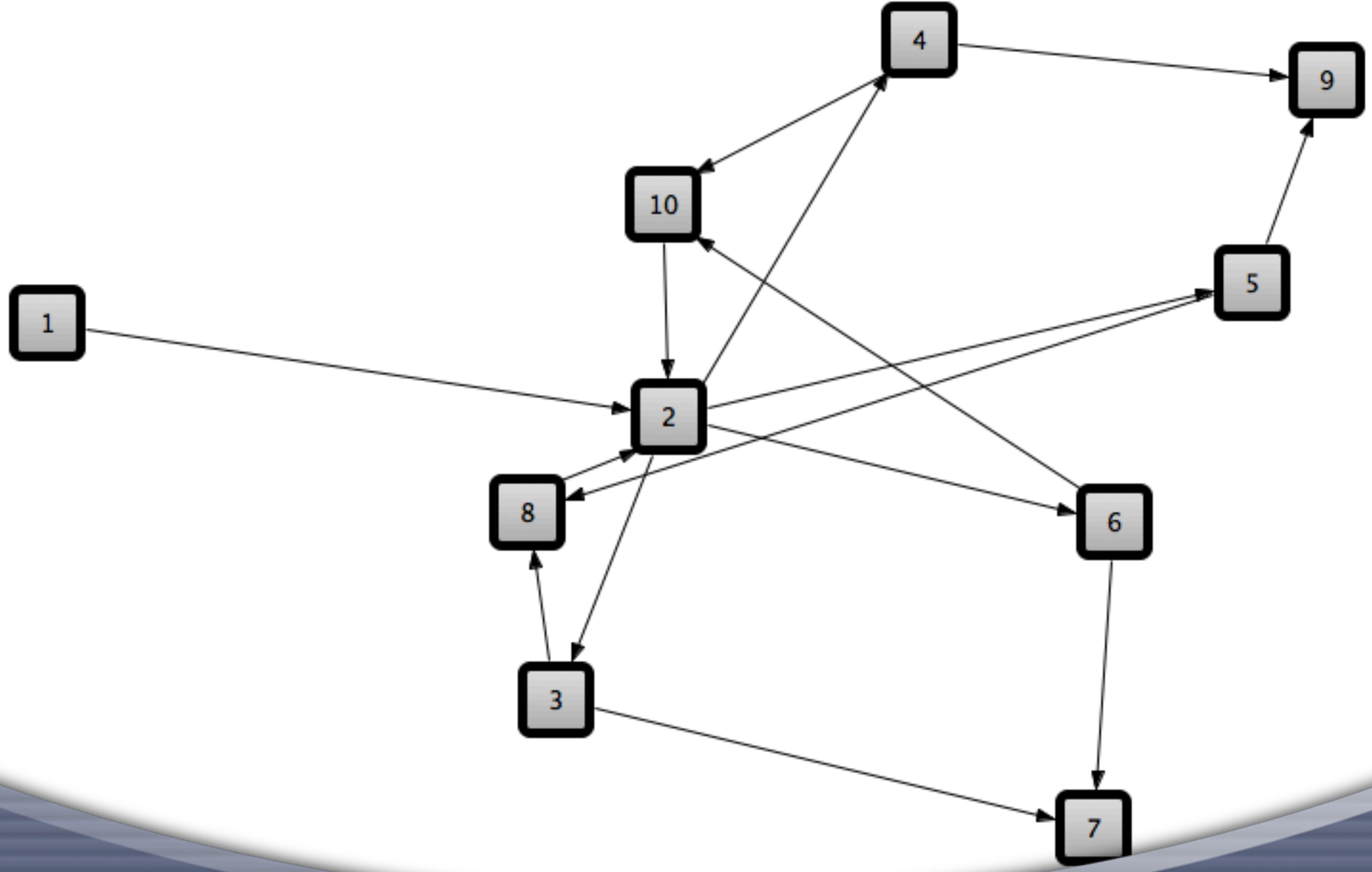
- ASAP automatically generates a template progress measure (much like queries)
- We just have to fill in the blanks
- Let's use the number of eating philosophers as the **progress value**

Example: Progress Measure

```
fun query (state, events) =
let
  fun query'New_Page { Waiting, Has_One, Eating,
                       Philosophers, Initialized,
                       Chopsticks } =
    List.length Eating
  fun query'state { New_Page} = query'New_Page New_Page
in
  query'state state
end
```

Demo: The Sweep-line Method

- Create new from template (05)
- Change safety-checker to use sweep-line method instead (06)
- Note no change at top level
- Run check
- Move up progress measure



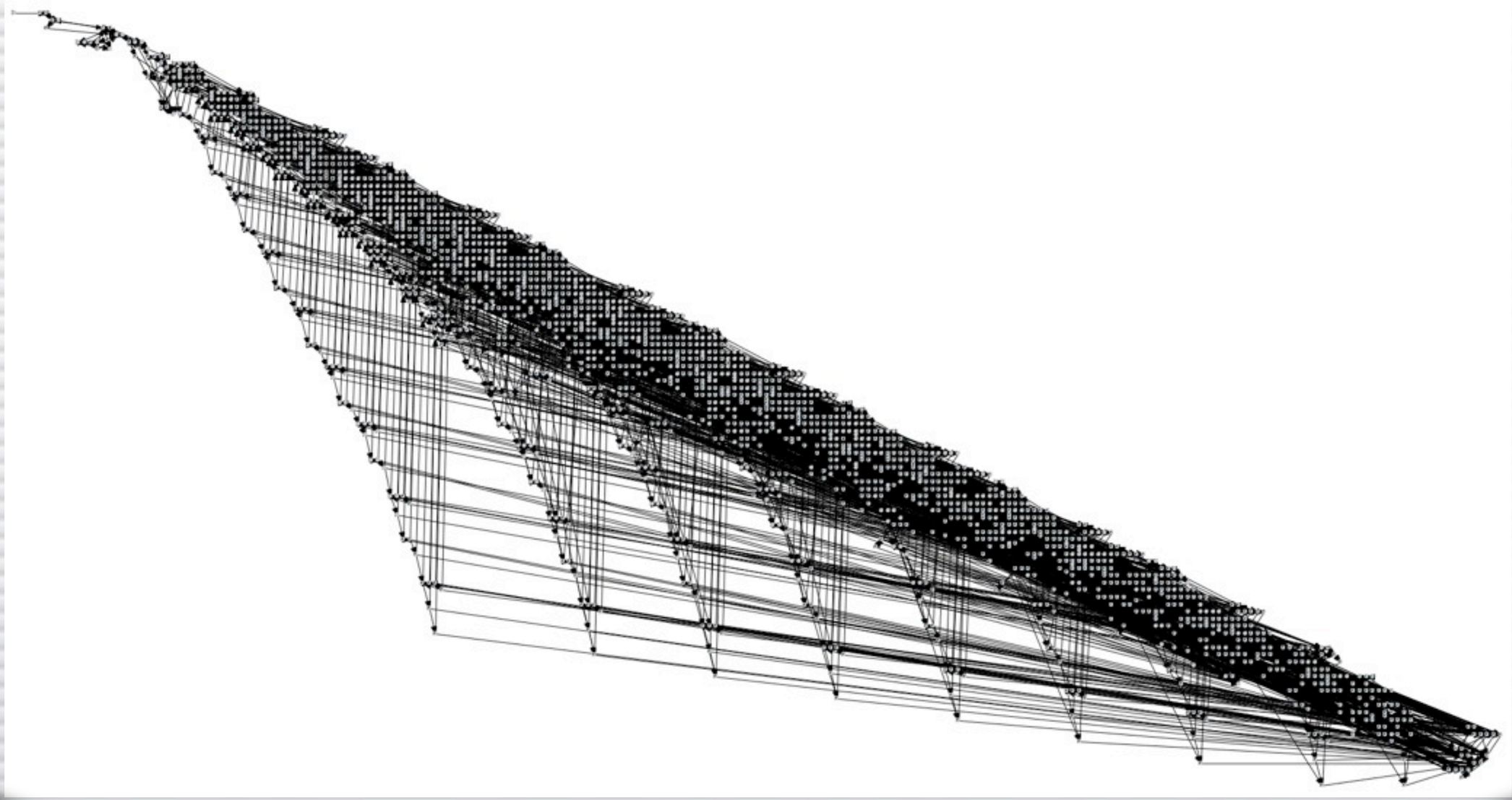
Example: Drawing SS Graphs

Drawing SS Graphs

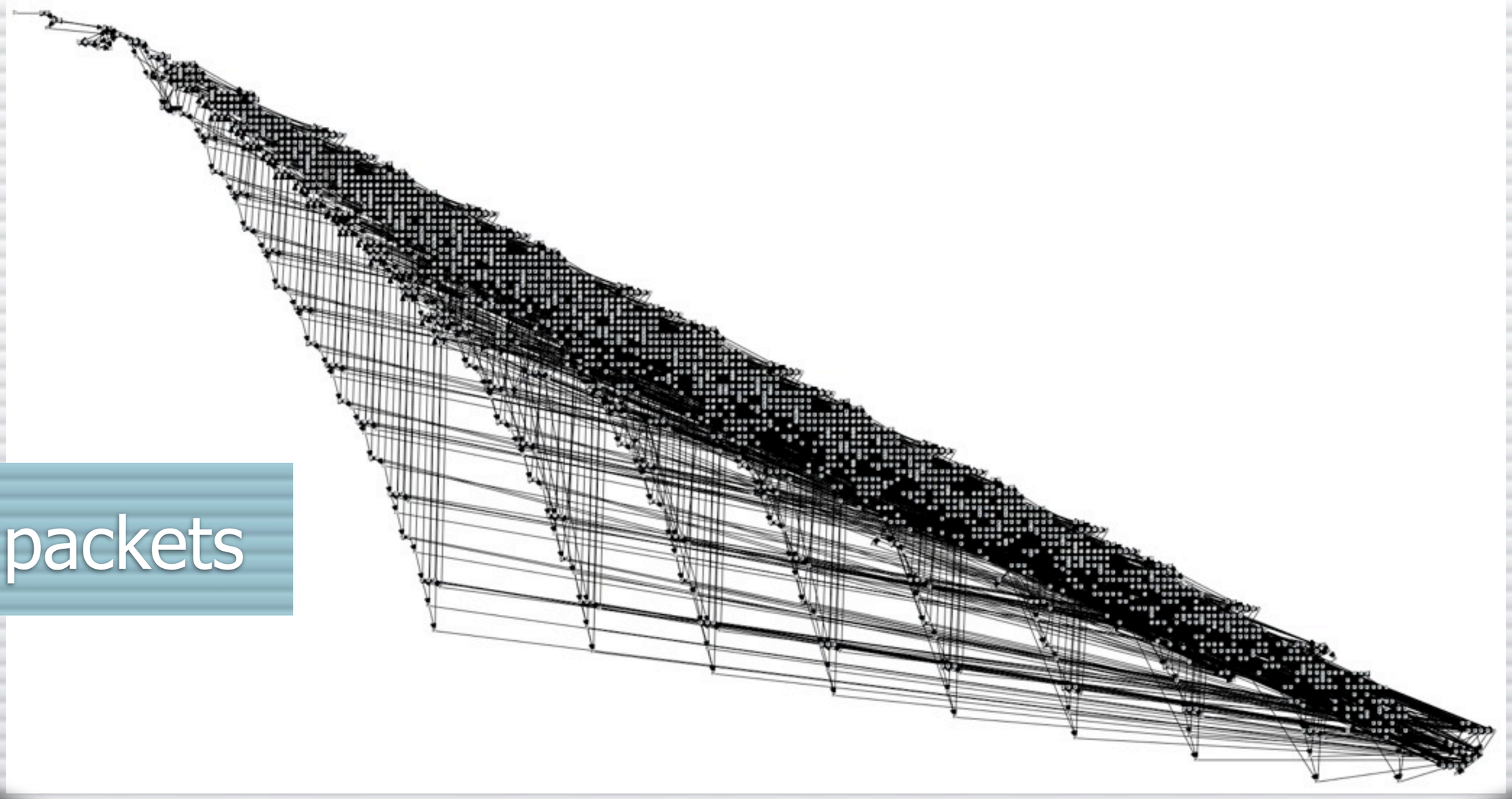
- CPN Tools supports interactive drawing of SS graphs
- ASAP supports automatic drawing of SS graphs
  - Only all of the graph (or predefined subsets)

Demo: Drawing SS Graphs (07)

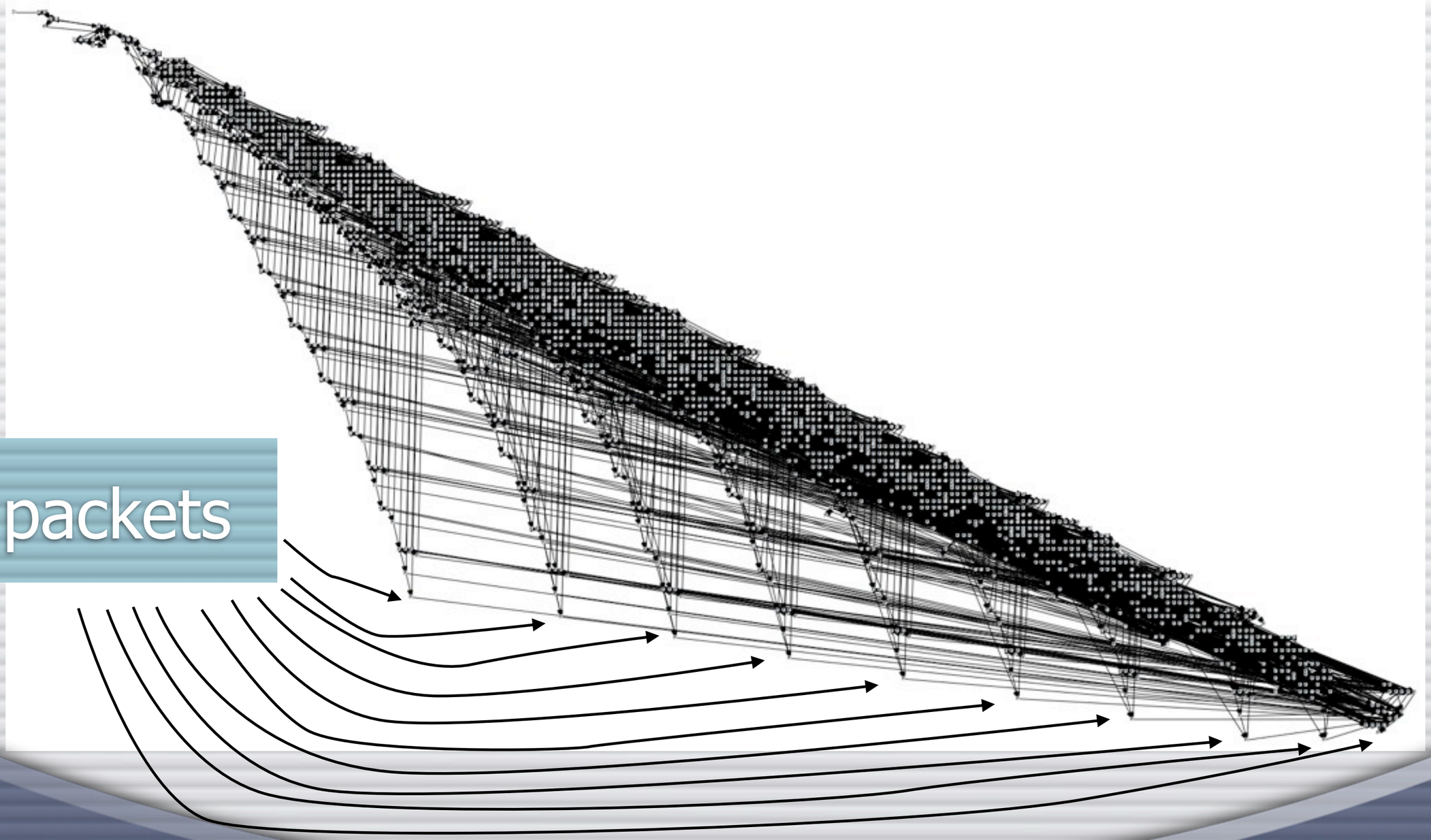
- Change safety checker to draw SS graph
- Change model size to 2 philosophers
- Play with layouts
- Export to DOT and GML



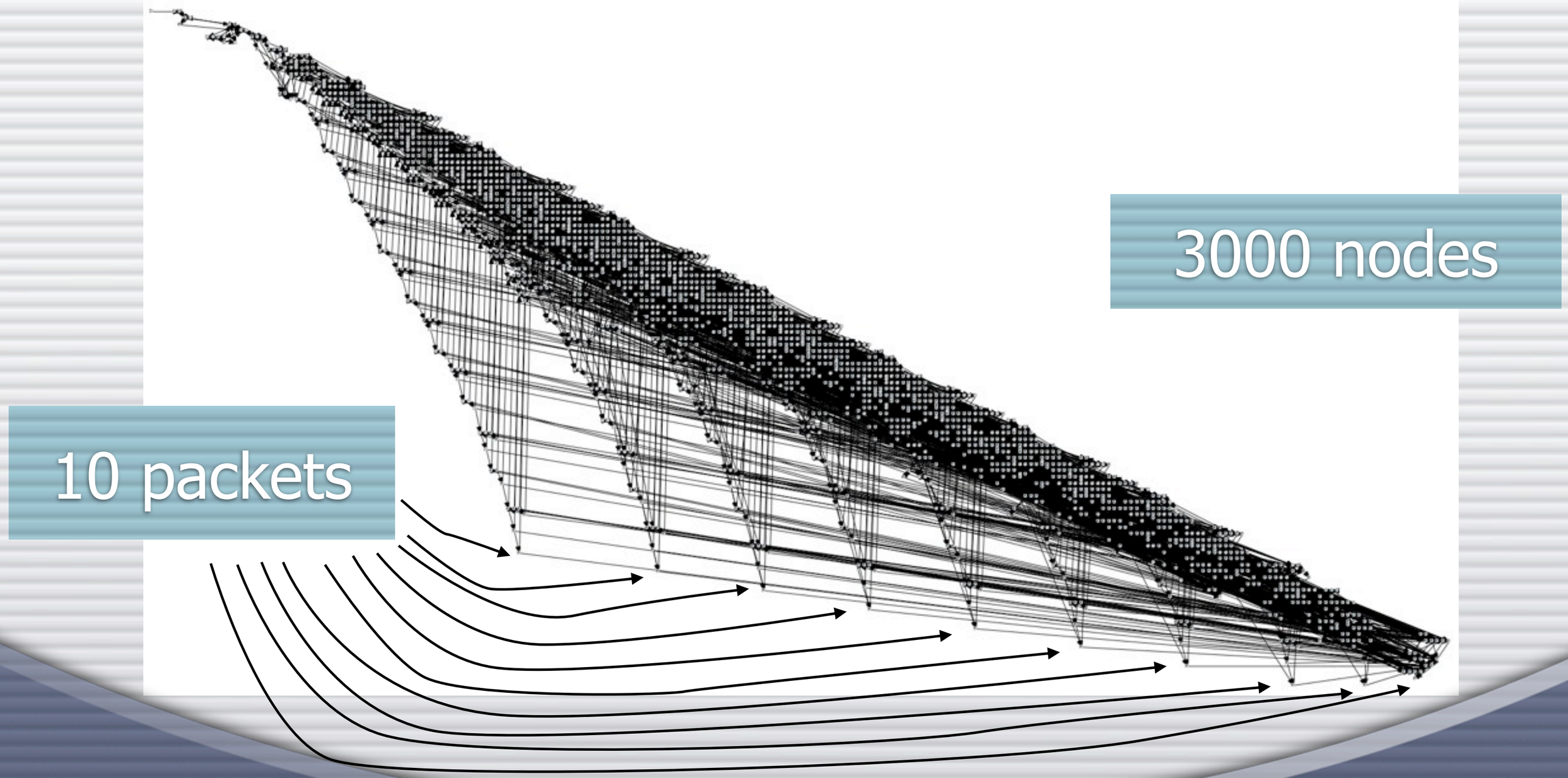
Example: Simple Protocol



Example: Simple Protocol



Example: Simple Protocol



Example: Simple Protocol