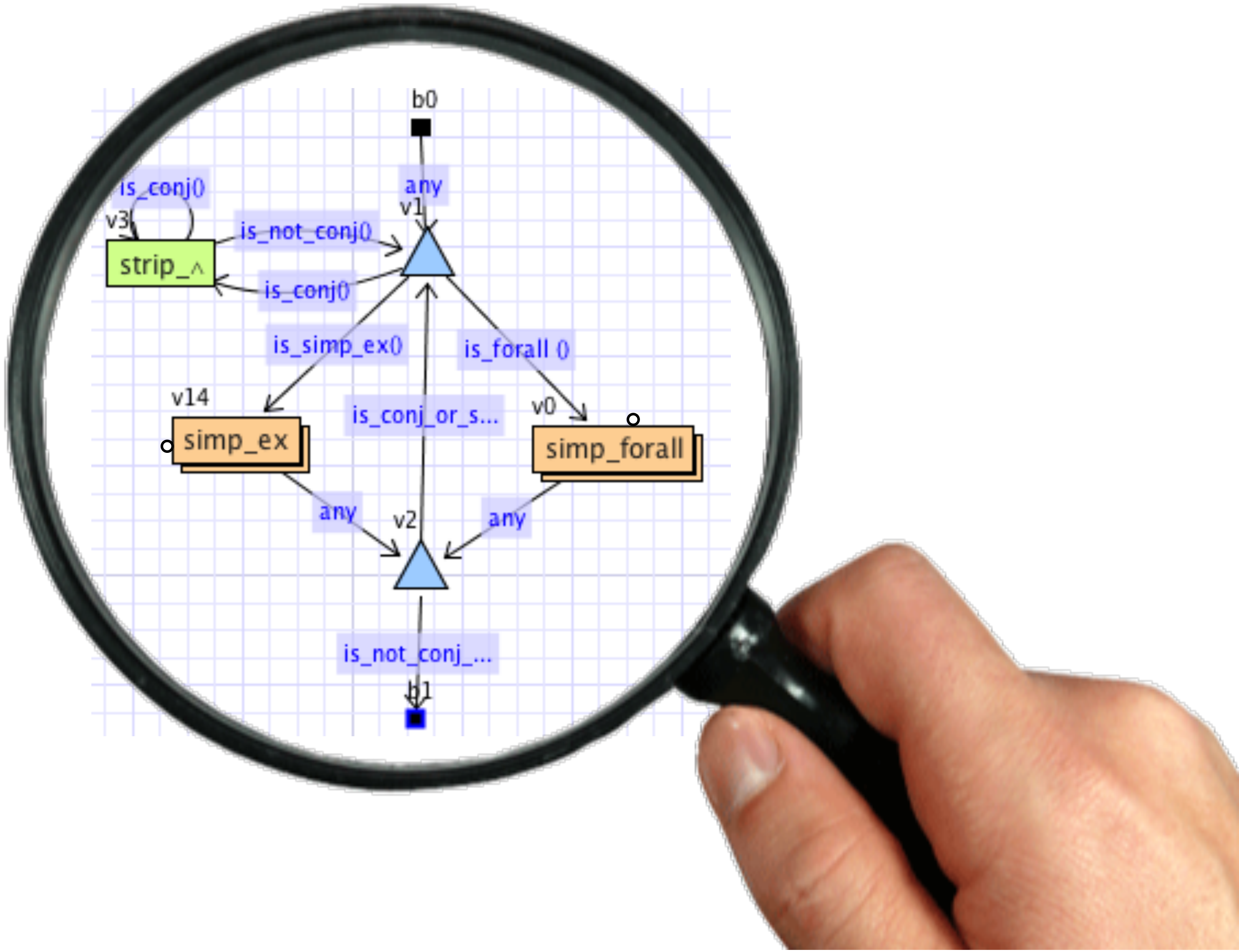
Developing & Debugging Proof Strategies

by TINKERING

Yuhui Lin,
Pierre Le Bras,
Gudmund Grov

April 2016, TACAS
Eindhoven



Stack-based strategies

LCF-style provers operate on open goals using **tactics**:

```
t: goal -> [goal]
```

Proof strategies are built from tactics using **tactical** combinators

Stack-based strategies

tac mytac := t_1 THEN t_2 THEN t_2 THEN t_3

mytac(g) :=

Stack-based strategies

tac mytac := t_1 THEN t_2 THEN t_2 THEN t_3

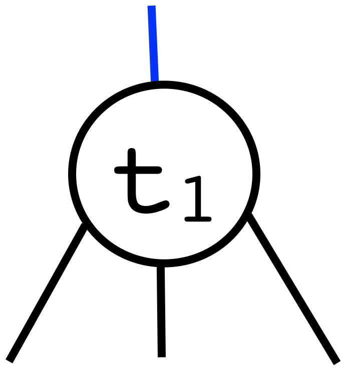
mytac(g) :=

Stack-based strategies

tac mytac := t_1 THEN t_2 THEN t_2 THEN t_3

↑

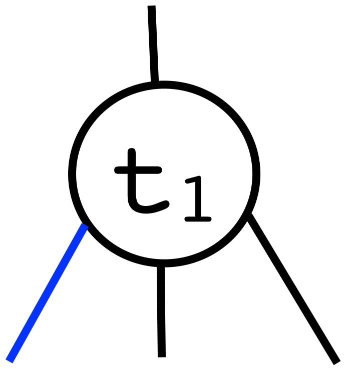
mytac(g) :=



Stack-based strategies

tac mytac := t_1 THEN $\underline{t_2}$ THEN t_2 THEN t_3

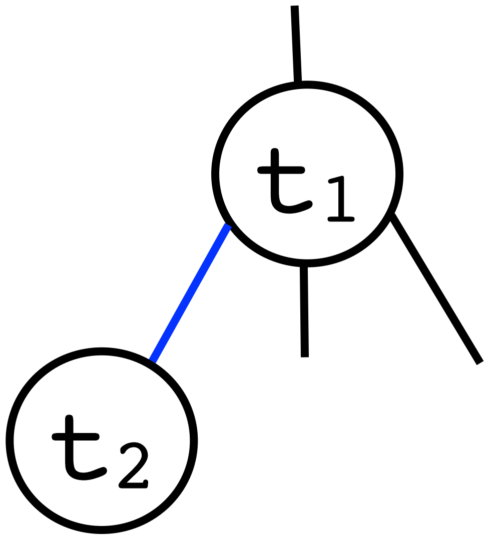
mytac(g) :=



Stack-based strategies

tac mytac := t_1 THEN t_2 THEN t_2 THEN t_3

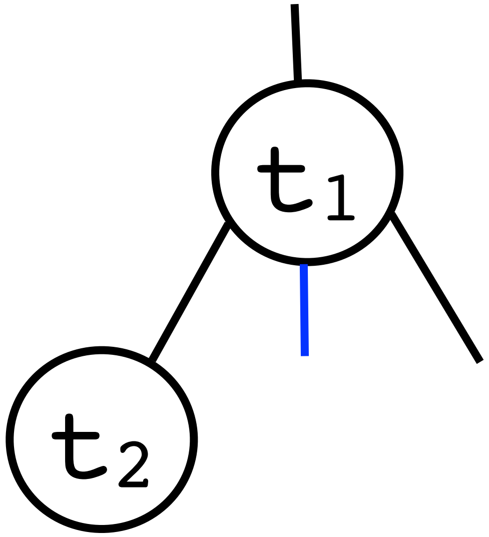
mytac(g) :=



Stack-based strategies

tac mytac := t_1 THEN t_2 THEN $\underline{t_2}$ THEN t_3

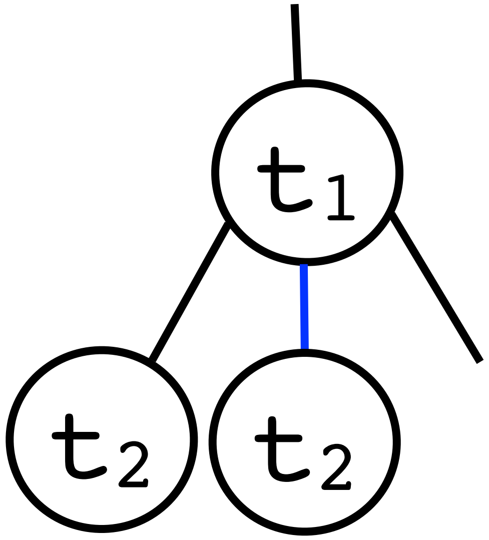
mytac(g) :=



Stack-based strategies

tac mytac := t_1 THEN t_2 THEN t_2 THEN t_3

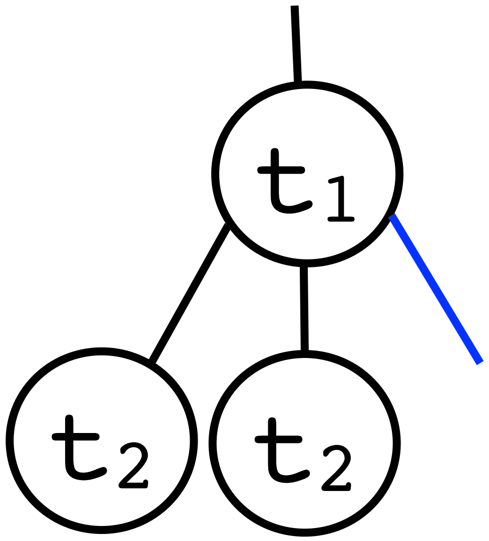
mytac(g) :=



Stack-based strategies

```
tac mytac := t₁ THEN t₂ THEN t₂ THEN t₃
```

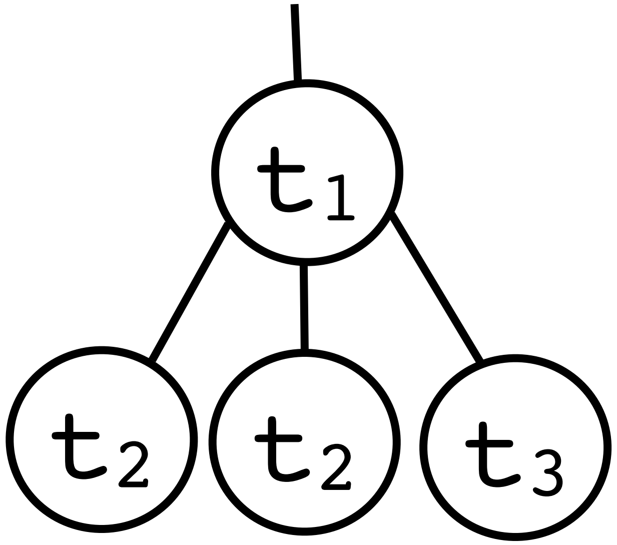
mytac(g) :=



Stack-based strategies

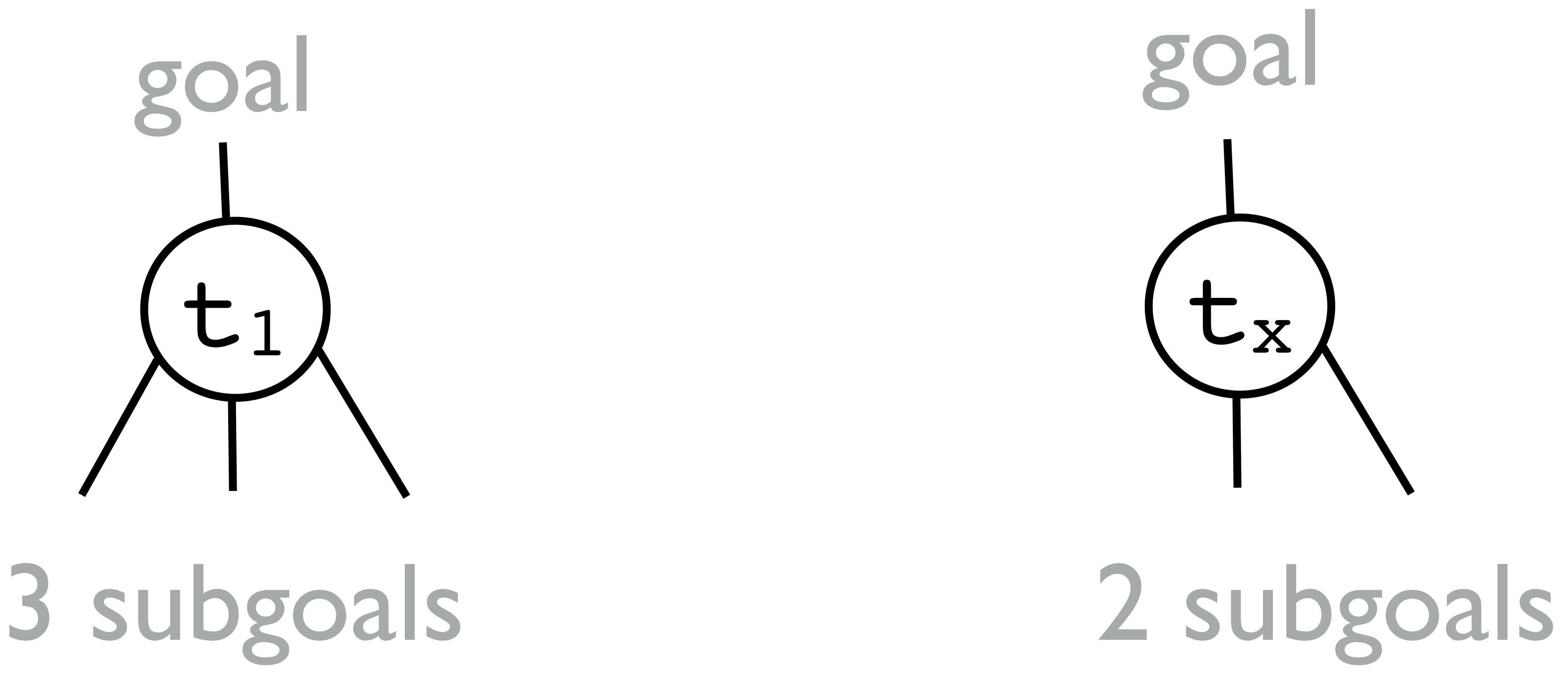
tac mytac := t_1 THEN t_2 THEN t_2 THEN t_3

mytac(g) :=



But sometimes it goes wrong....

Suppose we replace `t₁` with the "improved" tactic `tₓ`



Debugging

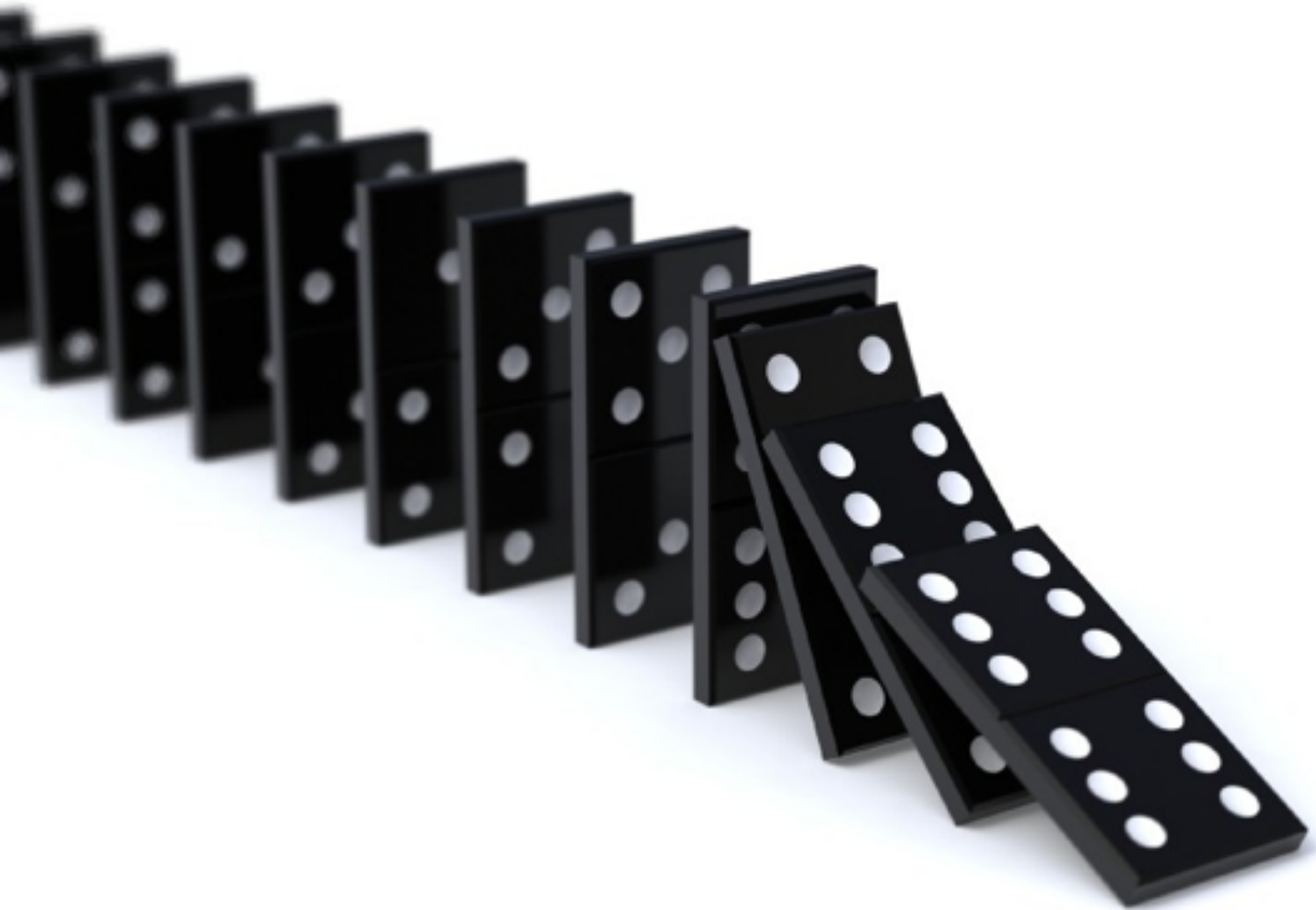
where did it go wrong?

actual error

here   or here

$$\mathbf{tac}\ \texttt{mytac := } \underline{t_x}\ \underline{\texttt{THEN } t_2}\ \texttt{THEN } \underline{t_2}\ \texttt{THEN } t_3$$

error



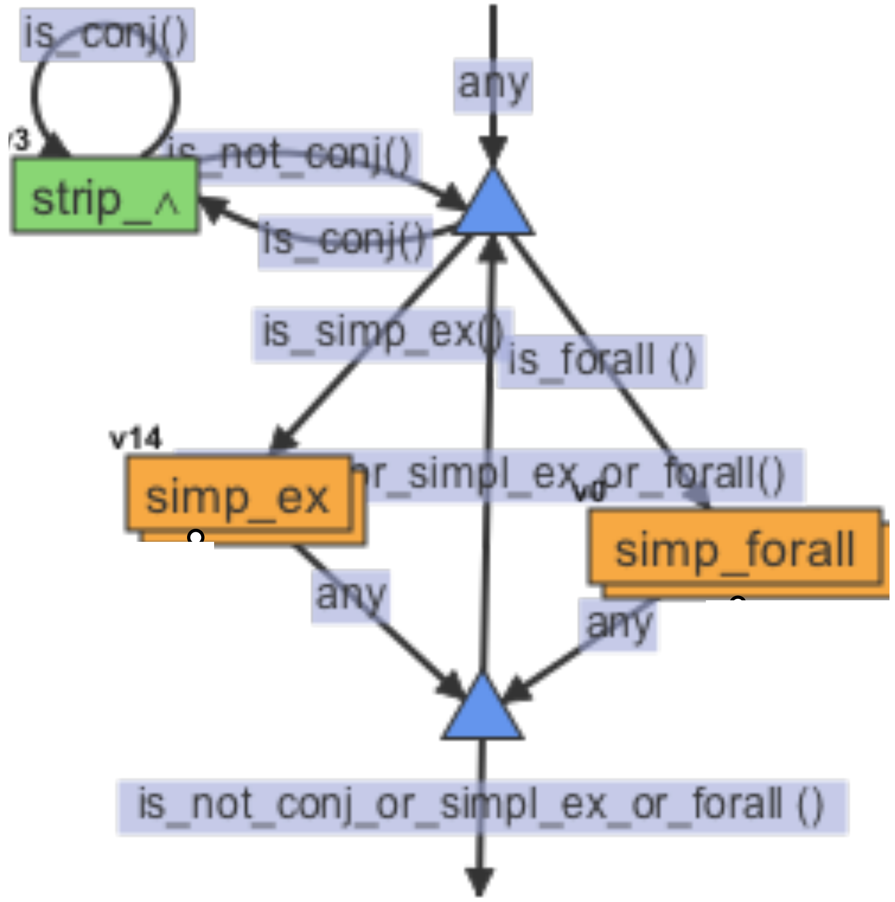
```
    (fn thm => rewrite_rule thms thm
        handle (Fail _) => thm)) o rev) THEN
    (TRY_T (rewrite_tac thms)) THEN
    REPEAT strip_tac THEN
    TRY_T all_var_elim_asm_tac THEN_TRY
    (z_quantifiers_elim_tac THEN
    (fn gl => let   val ciz = set_check_is_z false;
    val res = (EXTEND_PC_T1 "'mmp1" all_asm_fc_tac[] THEN
            (basic_res_tac2 3 [eq_refl_thm]
        ORELSE_T basic_res_tac3 3 [eq_refl_thm])) gl;
        val _ = set_check_is_z ciz; in res end

)));
```

error

```
fun z_basic_prove_tac (thms: THM list) : TACTIC = (
    TRY_T all_var_elim_asm_tac THEN
    DROP_ASMS_T (MAP_EVERY (strip_asm_tac o
    (fn thm => rewrite_rule thms thm
        handle (Fail _) => thm)) o rev) THEN
    (TRY_T (rewrite_tac thms)) THEN
    REPEAT strip_tac THEN
    TRY_T all_var_elim_asm_tac THEN_TRY
    (z_quantifiers_elim_tac THEN
    (fn gl => let   val ciz = set_check_is_z false;
          (basic_res_tac2 3 [eq_refl_thm]
        ORELSE_T basic_res_tac3 3 [eq_refl_thm])) gl;
        val _ = set_check_is_z ciz; in res end
    (fn thm => rewrite_rule thms thm
        handle (Fail _) => thm)) o rev) THEN
    (TRY_T (rewrite_tac thms)) THEN
    REPEAT strip_tac THEN
    TRY_T all_var_elim_asm_tac THEN_TRY
    (z_quantifiers_elim_tac THEN
    (fn gl => let   val ciz = set_check_is_z false;
          (basic_res_tac2 3 [eq_refl_thm]
        ORELSE_T basic_res_tac3 3 [eq_refl_thm])) gl;
        val _ = set_check_is_z ciz; in res end
    (fn thm => rewrite_rule thms thm
        handle (Fail _) => thm)) o rev) THEN
    (TRY_T (rewrite_tac thms)) THEN
    REPEAT strip_tac THEN
```

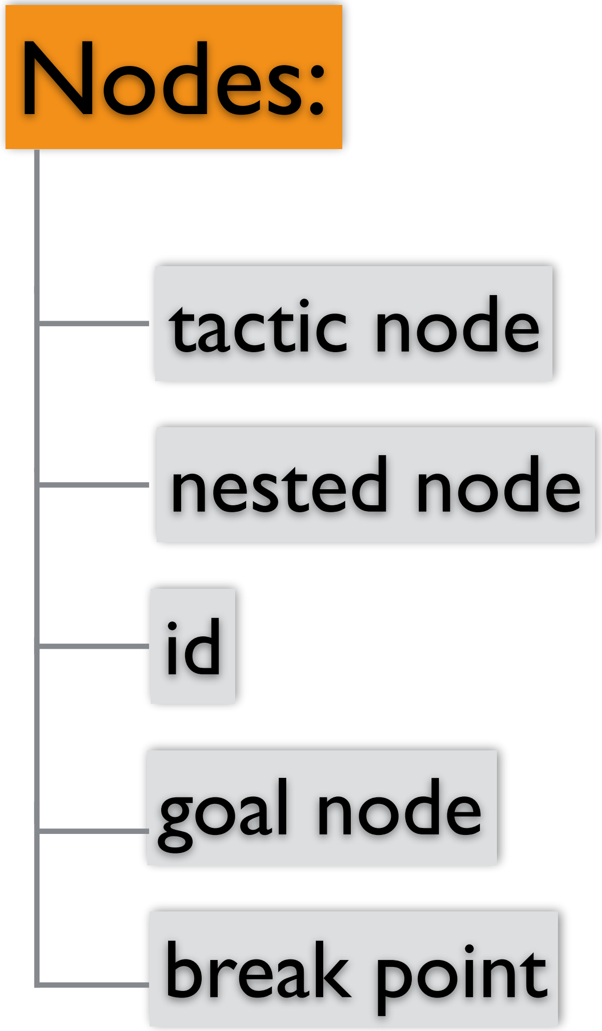
```
fun z_basic_prove_tac (thms: THM list) : TACTIC = (
        TRY_T all_var_elim_asm_tac THEN
        DROP_ASMS_T (MAP_EVERY (strip_asm_tac o
        (fn thm => rewrite_rule thms thm
            handle (Fail _) =>
        (TRY_T (rewrite_tac
        REPEAT strip_tac THE
        TRY_T all_var_elim_a
        (z_quantifiers_elim_ta
        (fn gl => let   val ciz =
              (basic_res_tac2
            ORELSE_T basic_                    ;
            val _ = set_check_is_z ciz; in res end
        (fn thm => rewrite_rule thms thm
            handle (Fail _) => thm)) o rev) THEN
        (TRY_T (rewrite_tac thms)) THEN
        REPEAT strip_tac THEN
        TRY_T all_var_elim_asm_tac THEN_TRY
        (z_quantifiers_elim_tac THEN
        (fn gl => let   val ciz = set_check_is_z false;
              (basic_res_tac2 3 [eq_refl_thm]
            ORELSE_T basic_res_tac3 3 [eq_refl_thm])) gl;
            val _ = set_check_is_z ciz; in res end
        (fn thm => rewrite_rule thms thm
            handle (Fail _) => thm)) o rev) THEN
        (TRY_T (rewrite_tac thms)) THEN
        REPEAT strip_tac THEN
```

Tinker



- proof strategies as hierarchical graphs
- tactic composition
  by connecting nodes with edges
- edges with goal types
- tactic application
  by consuming and producing goals
  through nodes

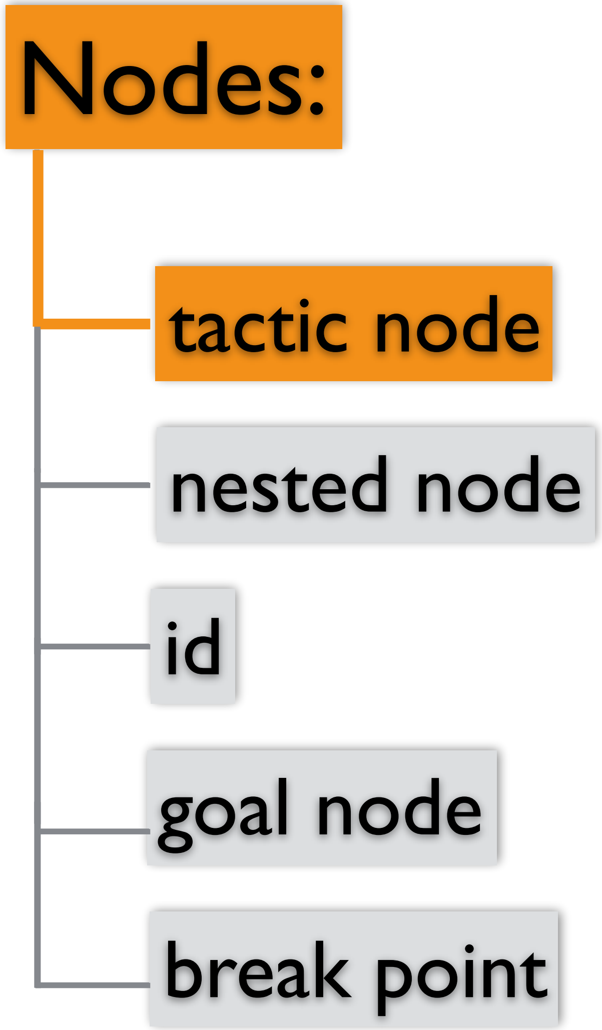
Tinker Tool



Nodes:

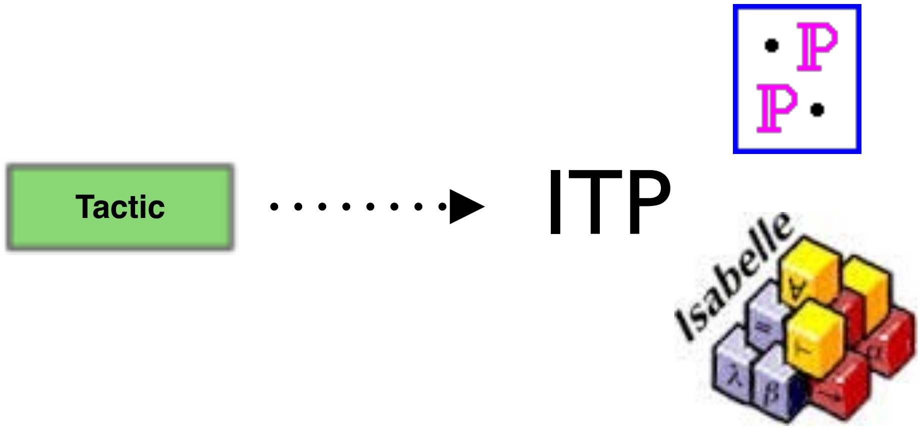
- tactic node
- nested node
- id
- goal node
- break point

Tinker Tool



Nodes:

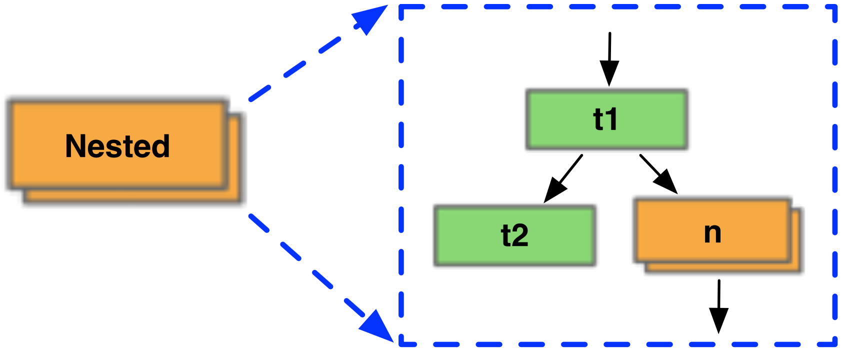
- tactic node
- nested node
- id
- goal node
- break point



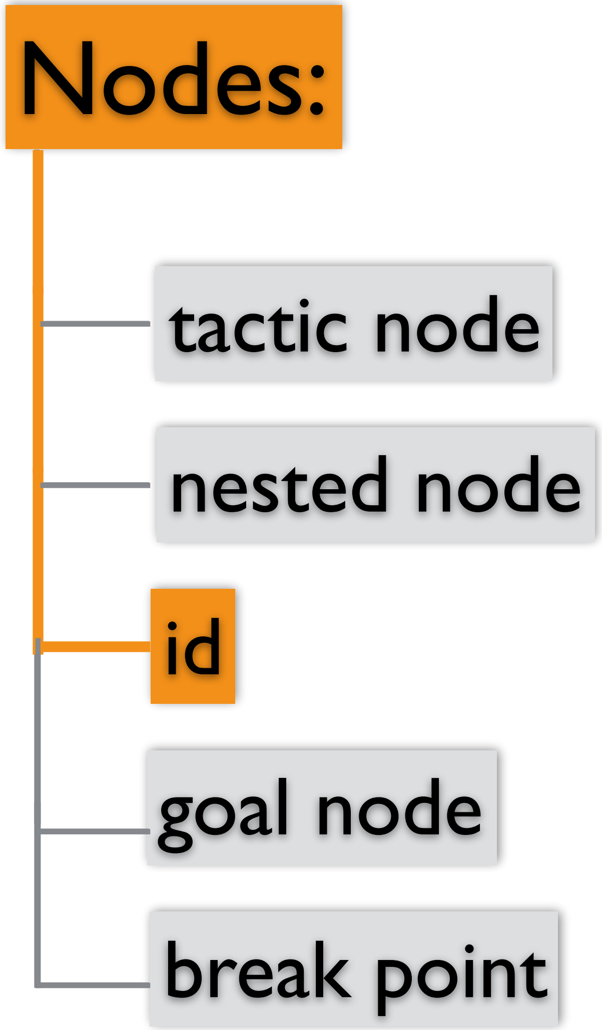
Tinker Tool

Nodes:

- tactic node
- **nested node**
- id
- goal node
- break point

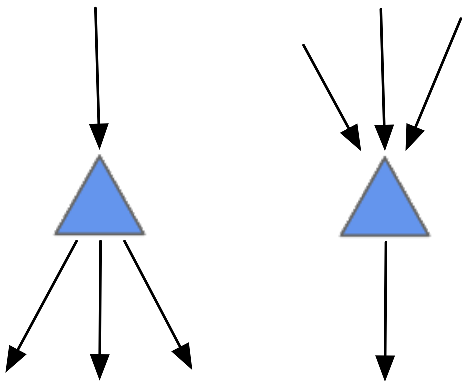


Tinker Tool

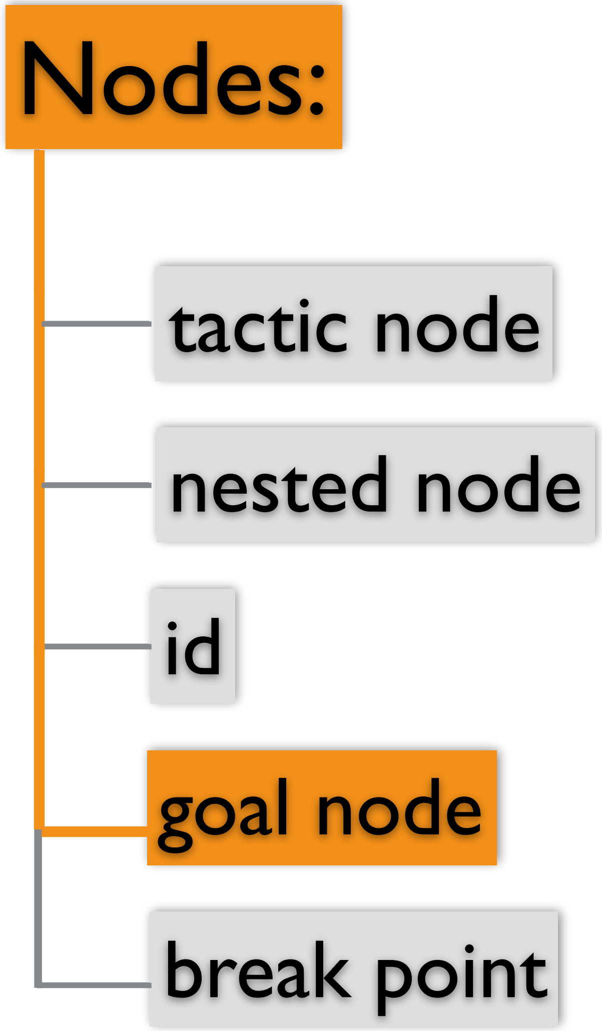


Nodes:

- tactic node
- nested node
- id
- goal node
- break point



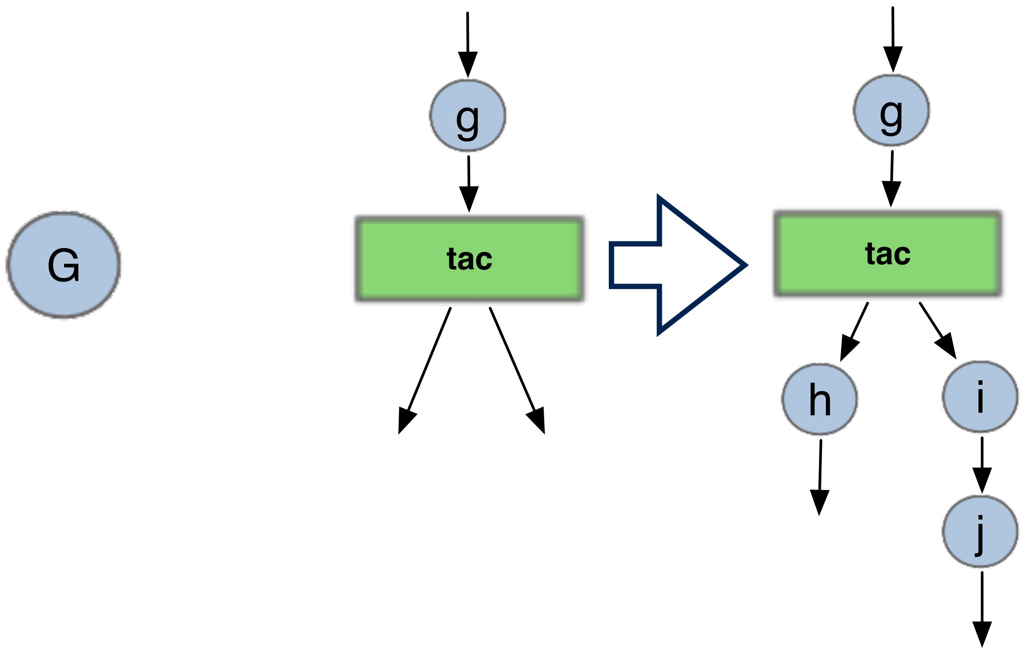
Tinker Tool



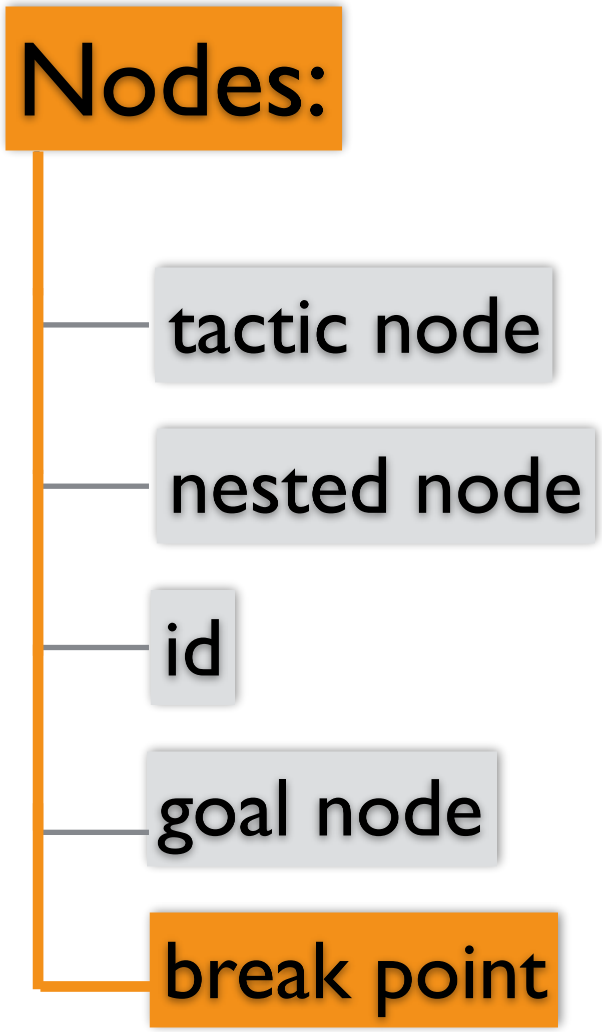
Nodes:

- tactic node
- nested node
- id
- goal node
- break point

`tac(g)=[h,i,j]`

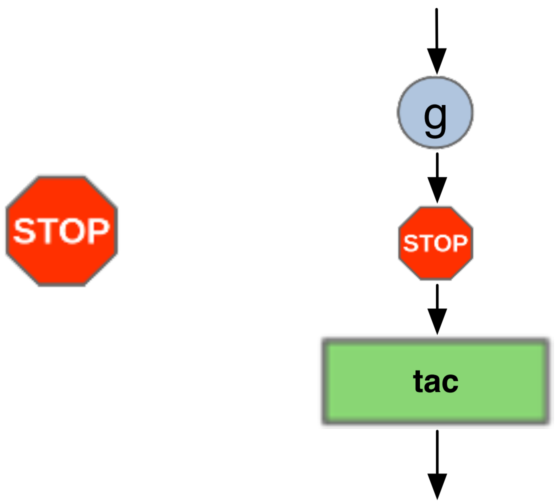


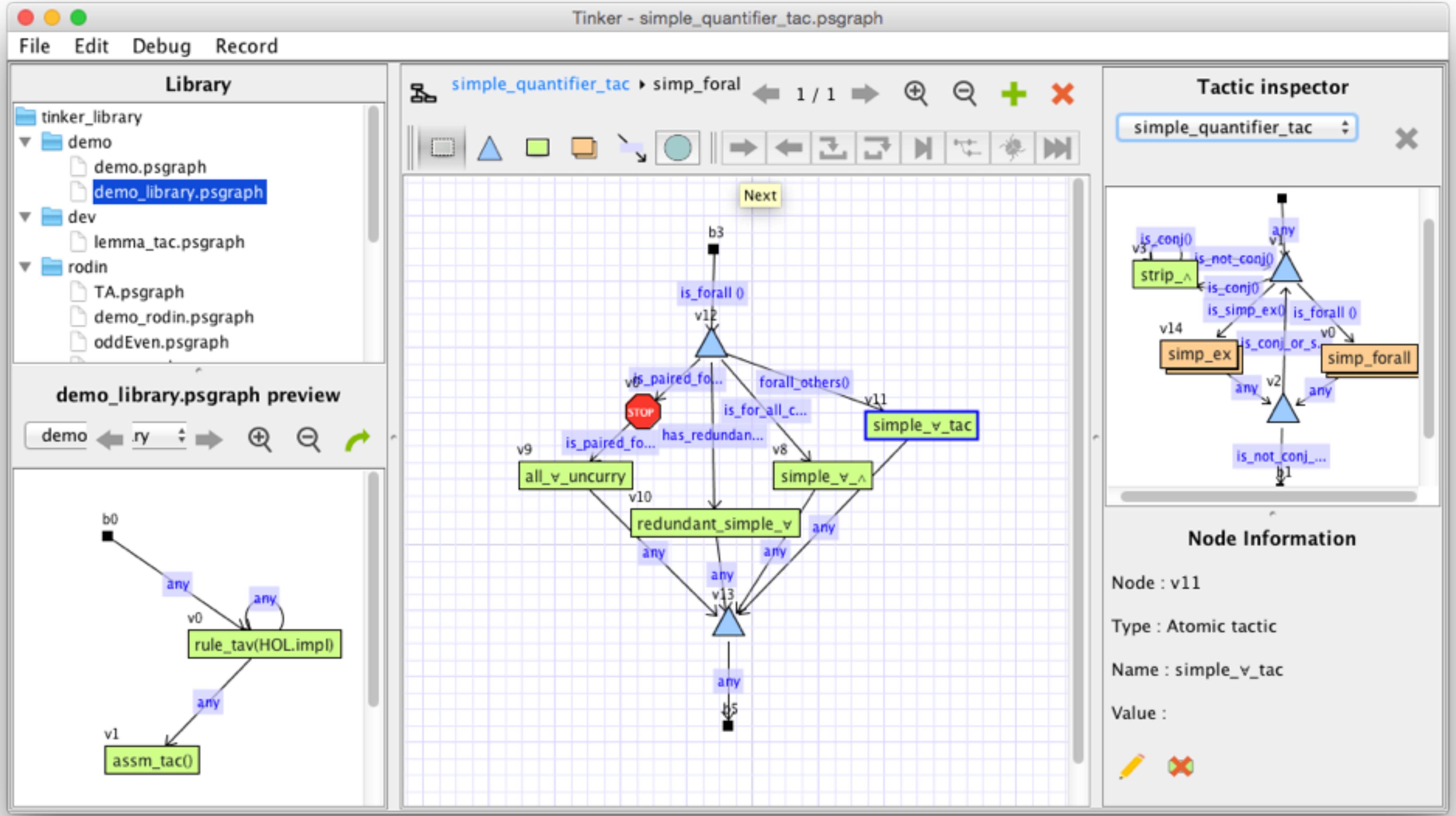
Tinker Tool



Nodes:

- tactic node
- nested node
- id
- goal node
- break point





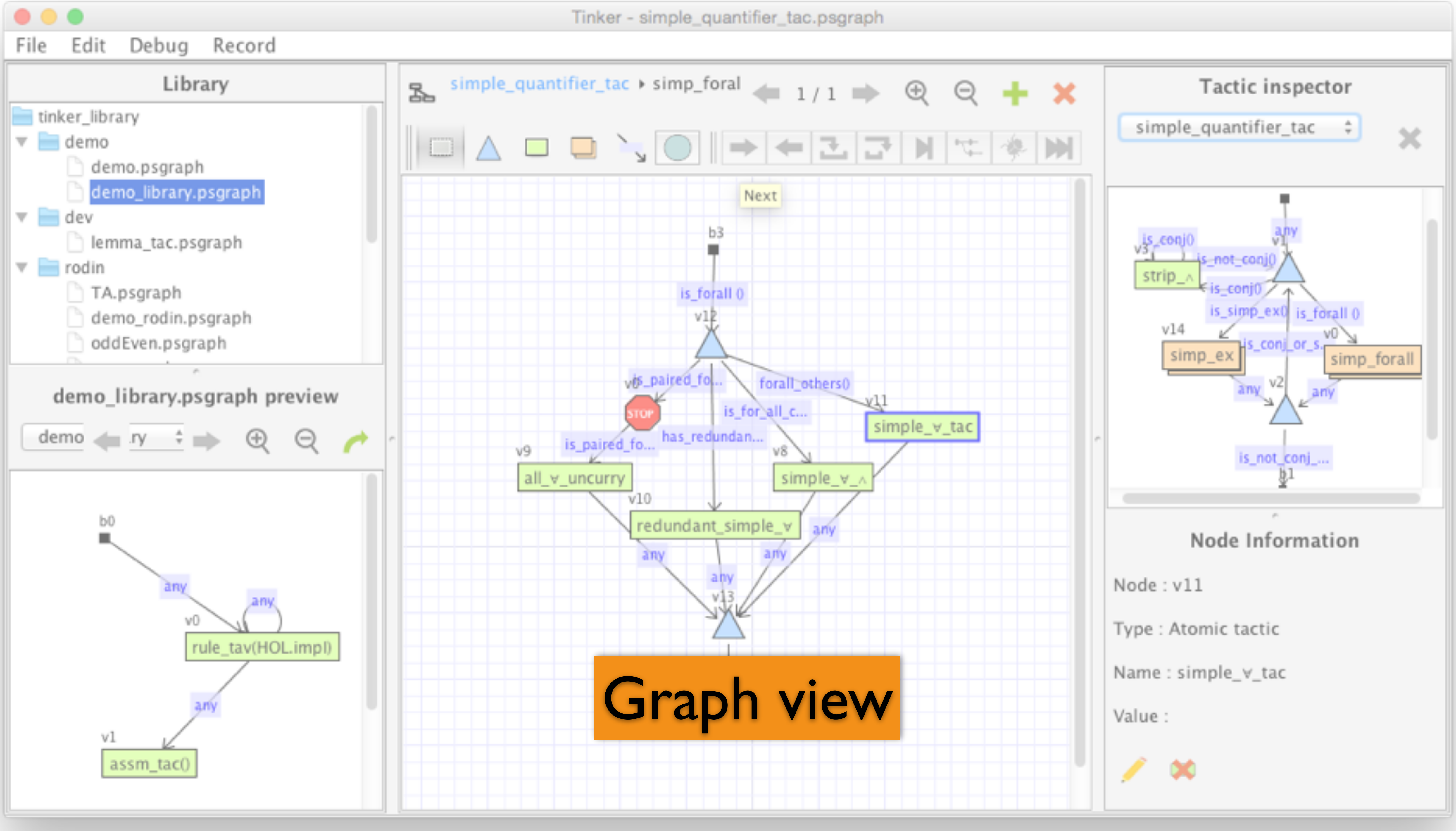
Node Information

Node : v11

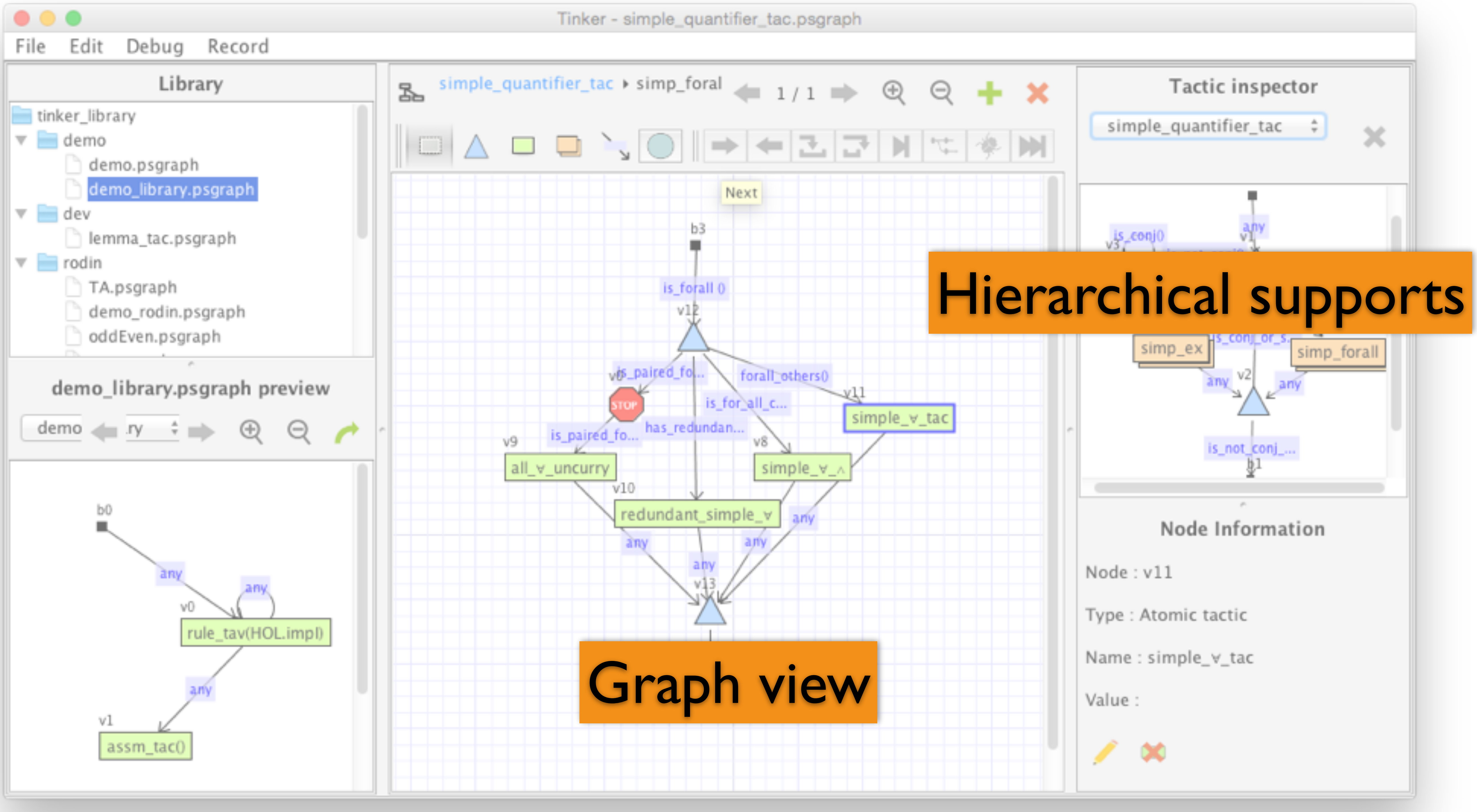
Type : Atomic tactic

Name : simple_∀_tac

Value :


Tinker - simple_quantifier_tac.psgraph
File
Edit
Debug
Record
Library
tinker_library
demo
demo.psgraph
demo_library.psgraph
dev
lemma_tac.psgraph
rodin
TA.psgraph
demo_rodin.psgraph
oddEven.psgraph
demo_library.psgraph preview
demo
ry
b0
any
any
v0
rule_tav(HOL.impl)
any
v1
assm_tac()
simple_quantifier_tac
simp_foral
1 / 1
Next
b3
is_forall ()
v12
is_paired_fo...
v6
STOP
forall_others()
v11
is_for_all_c...
simple_∀_tac
is_paired_fo...
has_redundan...
v9
v8
all_∀_uncurry
simple_∀_∧
v10
redundant_simple_∀
any
any
any
any
v13
Tactic inspector
simple_quantifier_tac
is_conj()
any
v3
v1
is_not_conj()
strip_∧
is_conj()
is_simp_ex()
is_forall ()
v14
is_conj_or_s...
v0
simp_ex
simp_forall
any
v2
any
is_not_conj_...
b1
Node Information
Node : v11
Type : Atomic tactic
Name : simple_∀_tac
Value :

Graph view



Hierarchical supports

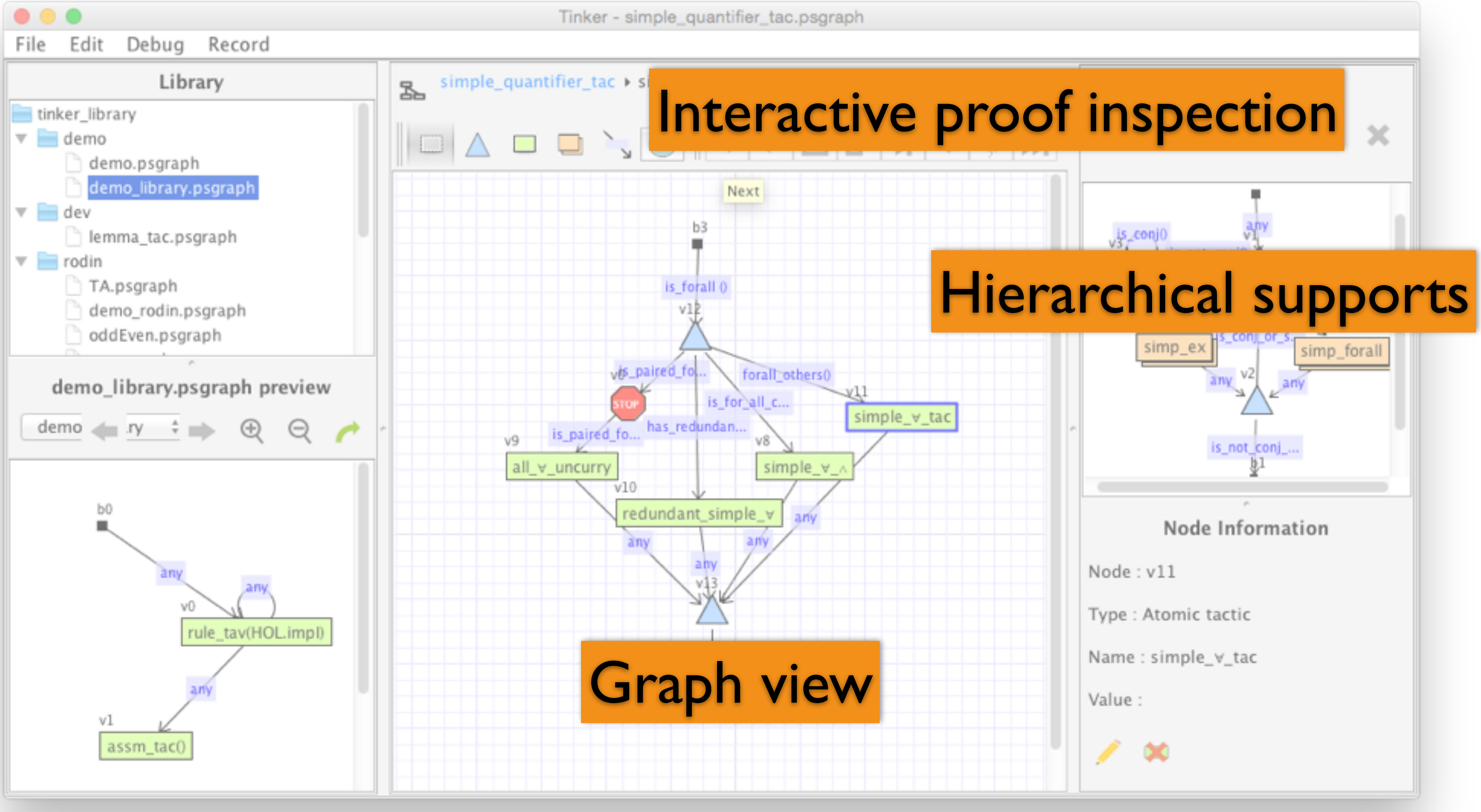
Graph view

Node : v11

Type : Atomic tactic

Name : simple_∀_tac

Value :



Node : v11

Type : Atomic tactic

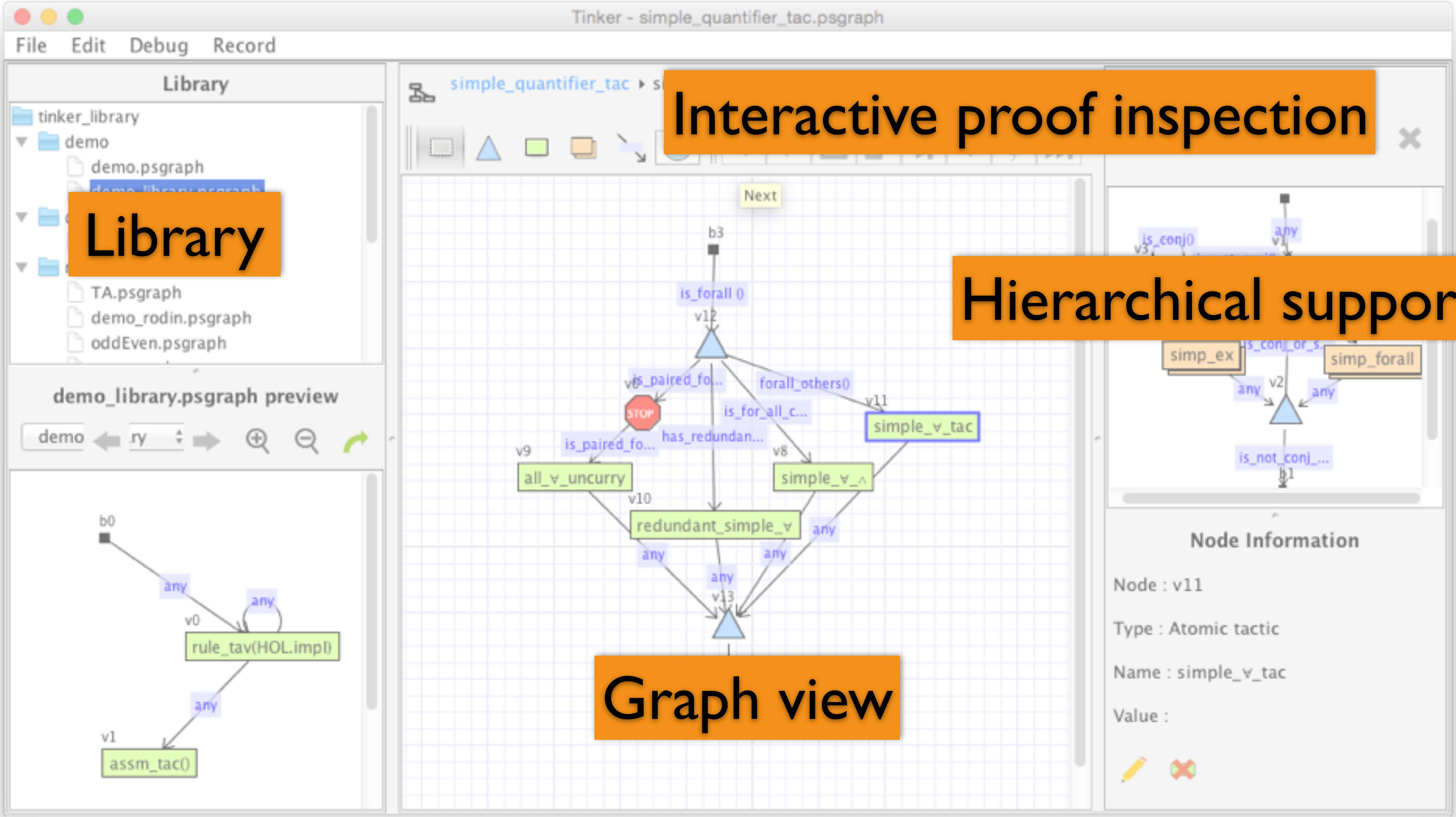
Name : simple_∀_tac

Value :

Interactive proof inspection

Hierarchical supports

Graph view



Node : v11

Type : Atomic tactic

Name : simple_∀_tac

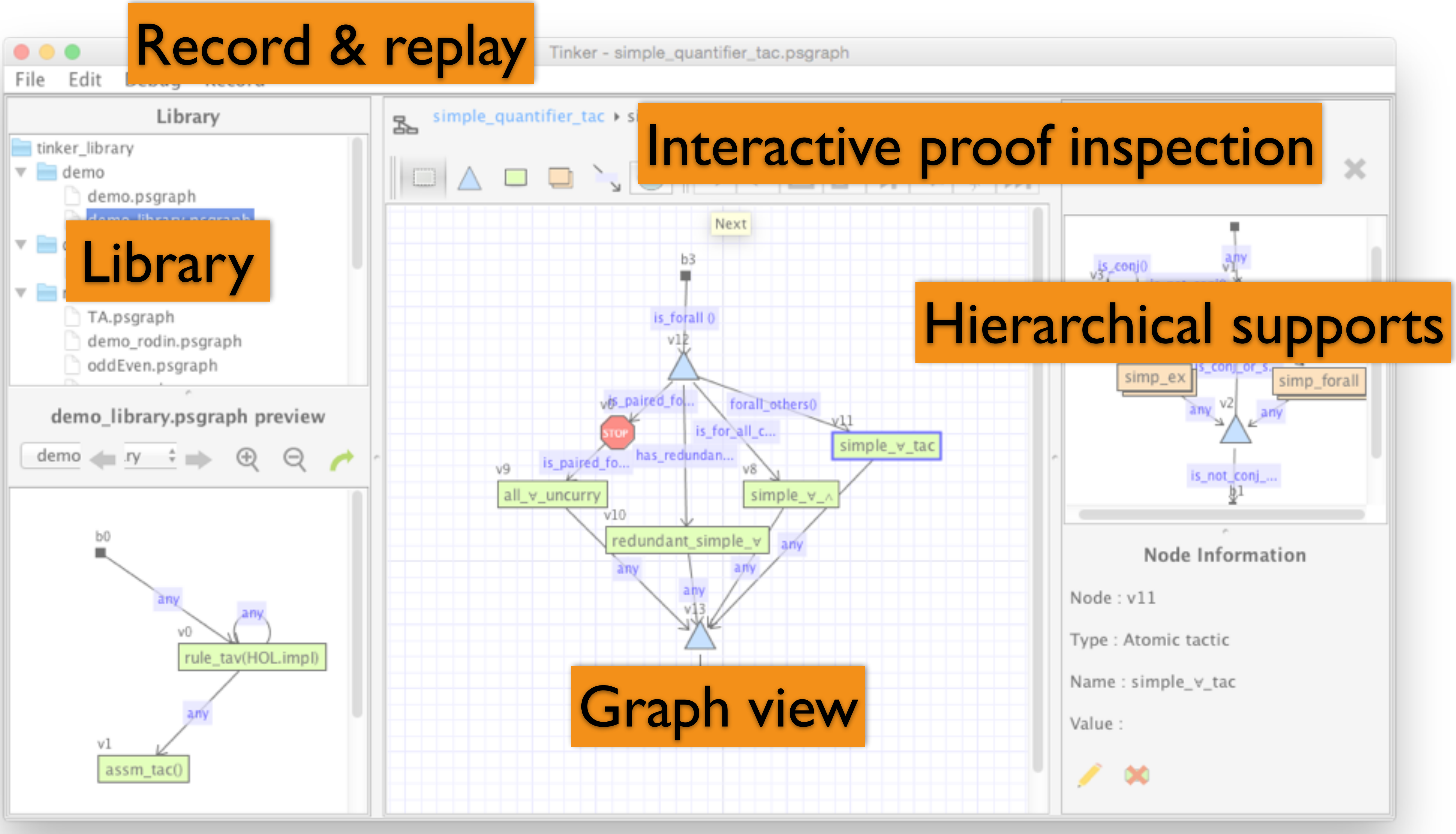
Value :

Interactive proof inspection

Library

Hierarchical supports

Graph view



Record & replay

Interactive proof inspection

Library

Hierarchical supports

Graph view

Tinker - simple_quantifier_tac.psgraph

Node Information

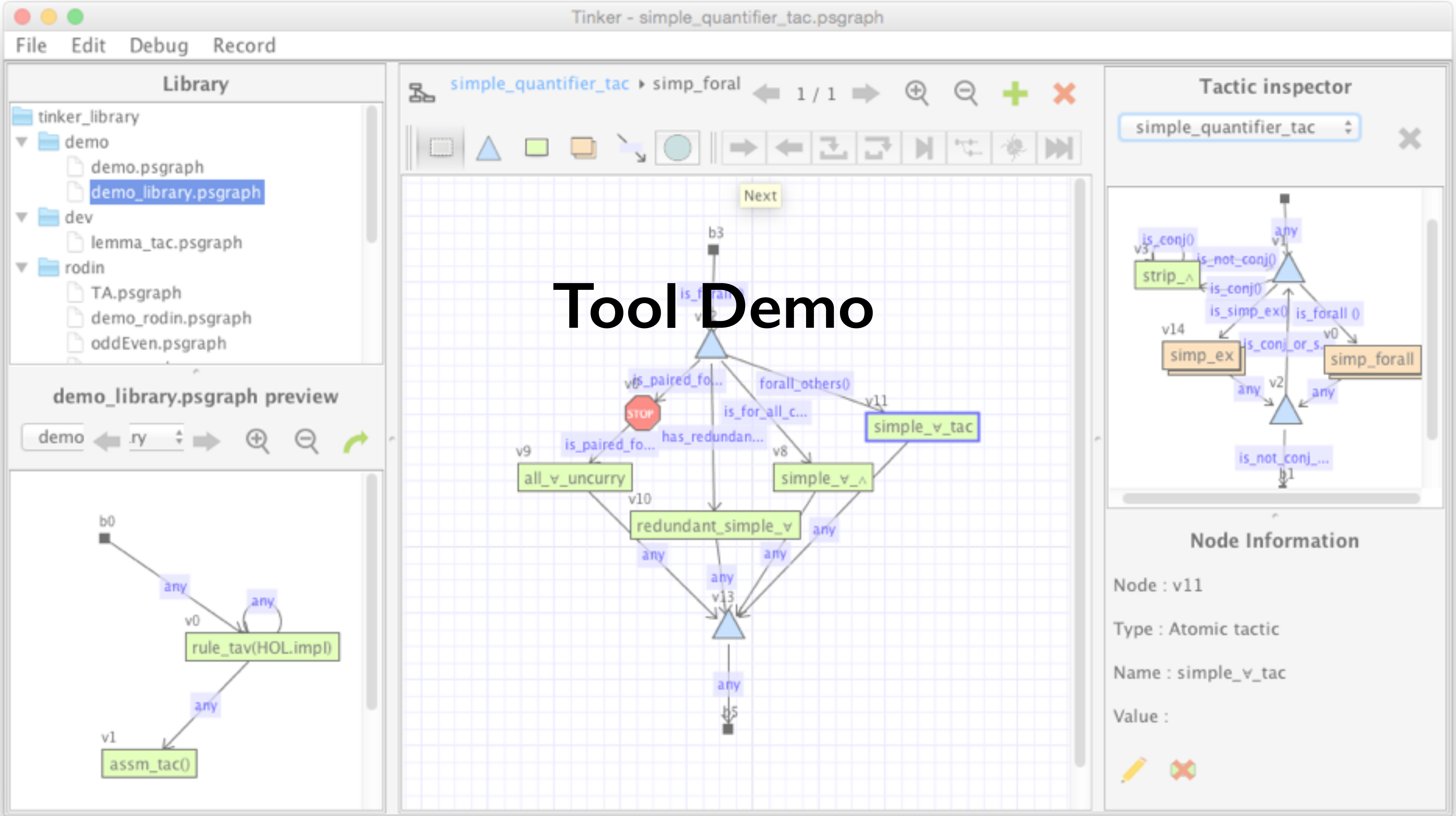
Node : v11

Type : Atomic tactic

Name : simple_∨_tac

Value :

Tool Demo



Node : v11

Type : Atomic tactic

Name : simple_∀_tac

Value :

http://ggrov.github.io/tinker/