Using Widenings/Narrowings in Data Flow Analyses

An introduction

Vesal Maynard Vojdani

`vesal@ut.ee`

Tartu University

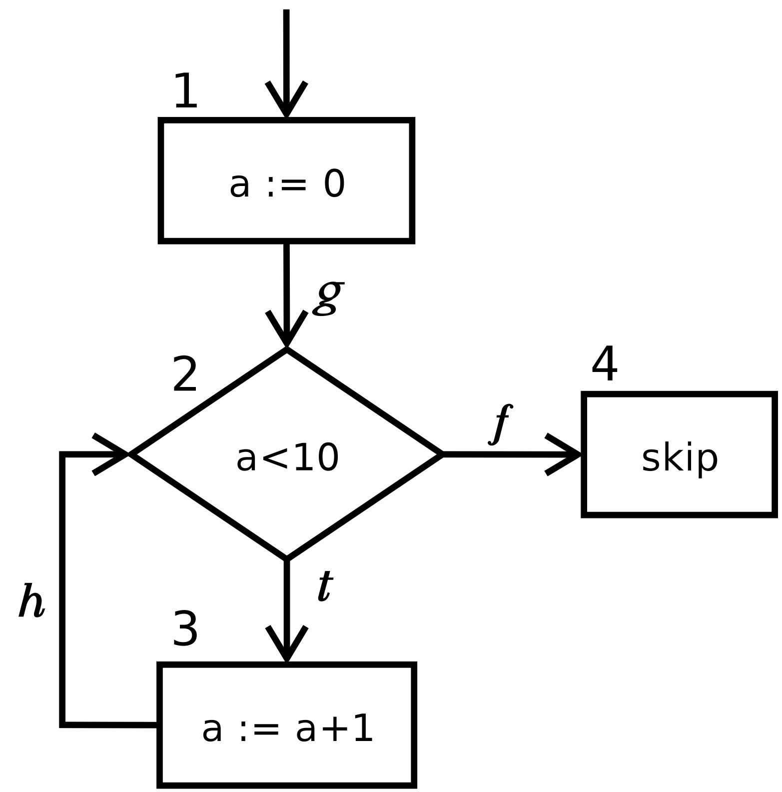
Overview

- Introduction $\longleftarrow$ you are here!
- Galois Connections
- From MOP to MFP
- Insufficiency of Galois connections
- Widenings/Narrowings
- Examples

Interval Analysis

- The goal is to do an interval analysis on the following simple program.

```
a := 0;
while a < 10 do
  a := a + 1;
```



Standard Semantics

- The standard semantics of a programming language defines how expression modify the state.
- The semantics is embodied in the transfer functions of the graph.
- The analysis must take into account
  - all possible input states
  - all possible executions

All possible inputs

Sets of States Semantics

- To analyse to program for all possible input states, we have to use the Set of States semantics.
- It is naturally derrived from the Standard semantics.
- Most precise domain, but impossible to use.
- Approximation is necessary.

The Lattice ordering

- Interesting properties are undecidable.
- Analyses must err on the safe side.
- The Lattice ordering: $x \sqsubseteq y$ means:
  - The set of programs satisfying x is included in the set of programs satisfying y.
  - The state y is a correct approximation of x.
  - The state x is more precise than y.

Galois connections

- Galois connection (more precisely adjunctions):

$$\alpha\colon X \to Y$$
$$\gamma\colon Y \to X$$

total functions that satisfy

$$\alpha(x) \sqsubseteq y \iff x \sqsubseteq \gamma(y)$$

Meaning of Galois connections

$$\alpha(x) \sqsubseteq y \iff x \sqsubseteq \gamma(y)$$

- $\alpha(x)$ is the most precise approximation of x.
- $\gamma(y)$ is the most general element, which can be soundly approximated by y.

Moving to the interval domain

- We can now abstract
  - Sets of State Semantics (visualize as points in space)
  - Collecting Semantics (abstract with projection, concretize as grid)
  - Interval domain (project to interval, concreetizes to rectangle)
- The transfer functions are induced by our abstraction:

$$f' = \alpha \circ f \circ \gamma$$

All possible executions

Notation

- The Control Flow Graph $G = (N, E, s)$, where
  - N is the set of nodes.
  - $E = N \times N$
  - s is the initial node.
- The analysis is an annotation $N \rightarrow D$.
- The transfer functions $tf : E \rightarrow (D \rightarrow D)$
- ι is the initial state.

Merge Over all Paths

Definition 1. The path semantics $[\![\pi]\!]_{tf}$ is simply the composition of the transfer functions along that path

$$[\![\varepsilon]\!]_{tf} = id_{D \to D}$$
$$[\![e_1, \ldots, e_n]\!]_{tf} = [\![e_2, \ldots, e_n]\!]_{tf} \circ tf(e_1)$$

Definition 2. The MOP solution to the data flow problem is defi ned at each node by

$$\mathsf{MOP}(n) = \bigsqcup \left\{ [\![\pi]\!]_{tf}\,(\iota) \mid \pi \text{ is a path from } s \text{ to } n \right\}$$

Least Fixed-point

MOP is not calculable.

Definition 3. The least (minimal) fixed point $\mathsf{MFP}(n)$ is the least solution to the following system:

$$\mathsf{MFP}(n) = \begin{cases} \iota & \text{if } n = s \\ \bigsqcup \{ tf(e)(\mathsf{MFP}(n')) \mid e = (n', n) \in E \} & \text{otherwise} \end{cases}$$

This is correct with respect to MOP, i.e. $\forall n$: $\mathsf{MOP}(n) \sqsubseteq \mathsf{MFP}(n)$.

PAG

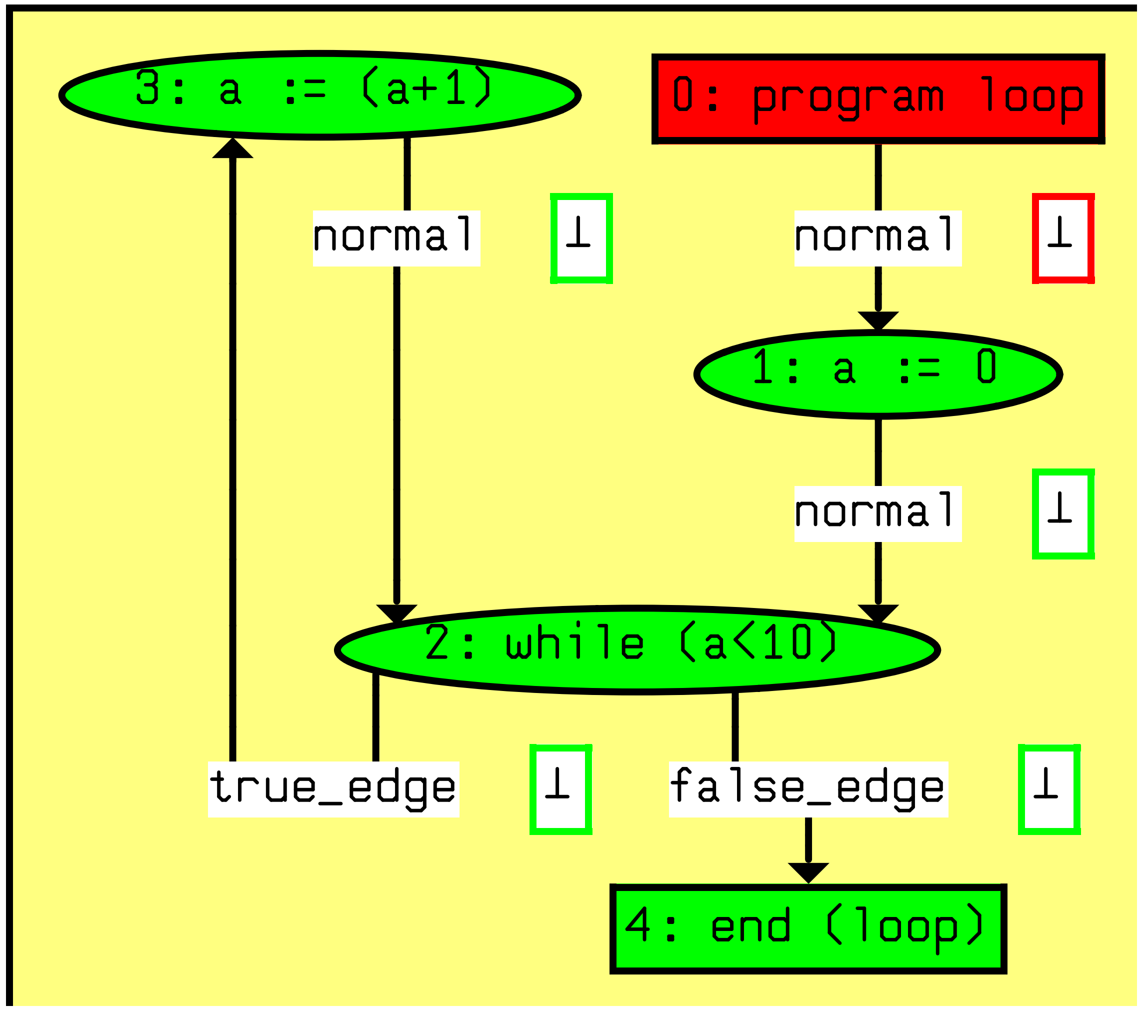
- The analysis is implemented in PAG – a Program Analysis Generator. See www.absint.com for further information.
- The output represents functions.

$$f(x) = \begin{cases} (0, 10) & \text{if } x = \texttt{a} \\ (-\infty, \infty) & \text{otherwise} \end{cases}$$

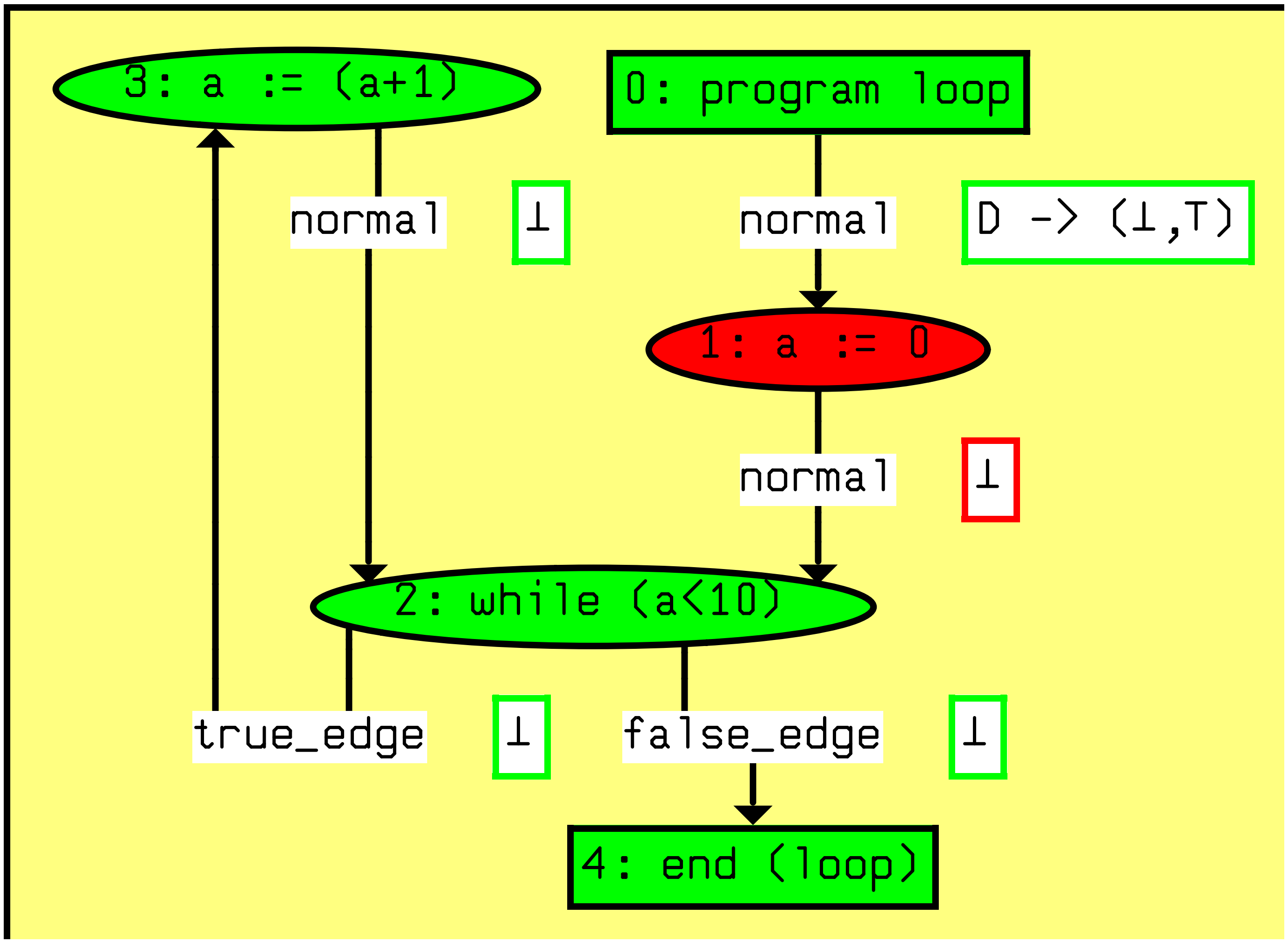
is written as

$$D \rightarrow (\perp, \top)$$
$$0 \rightarrow (0, 10)$$

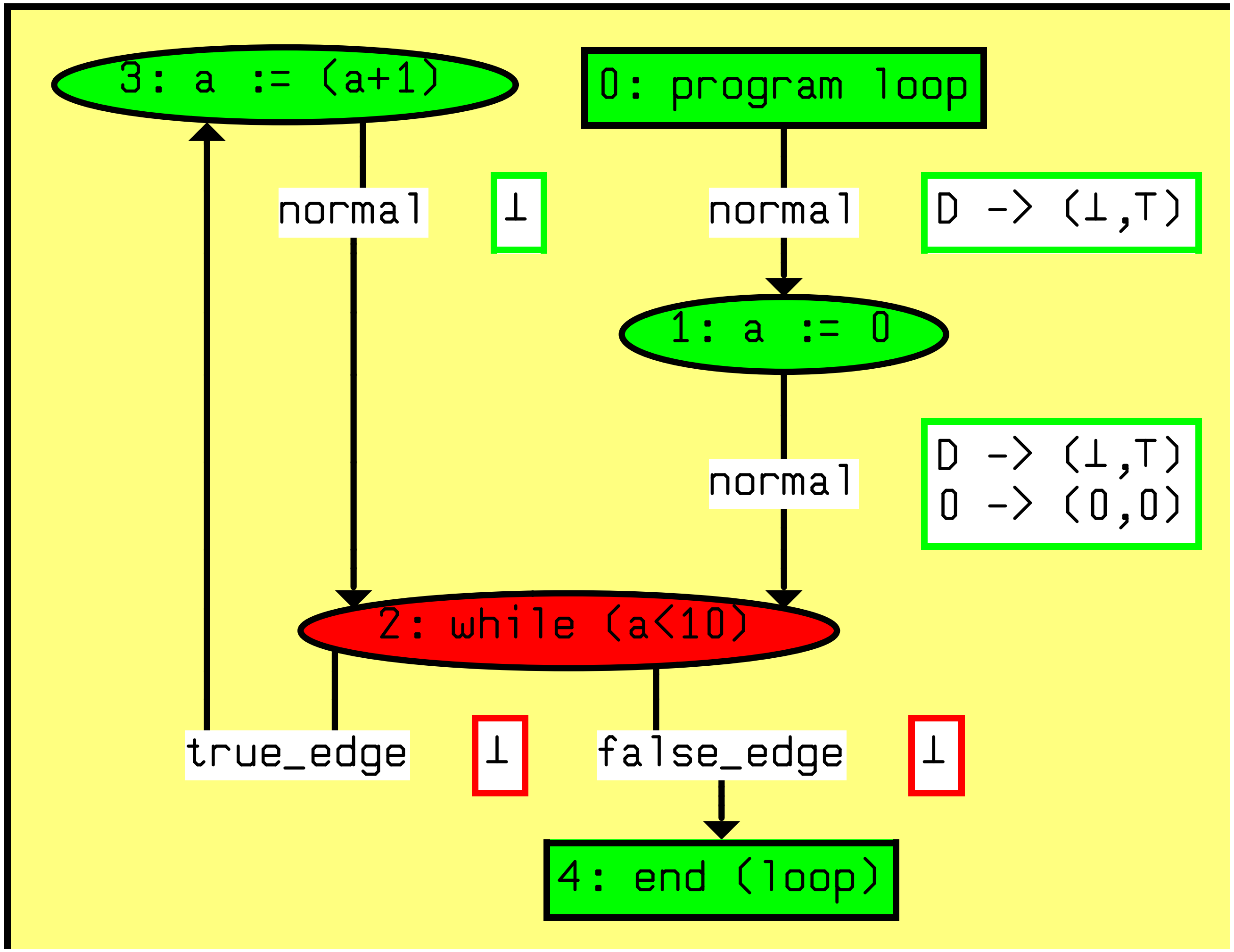
Testing the Analysis



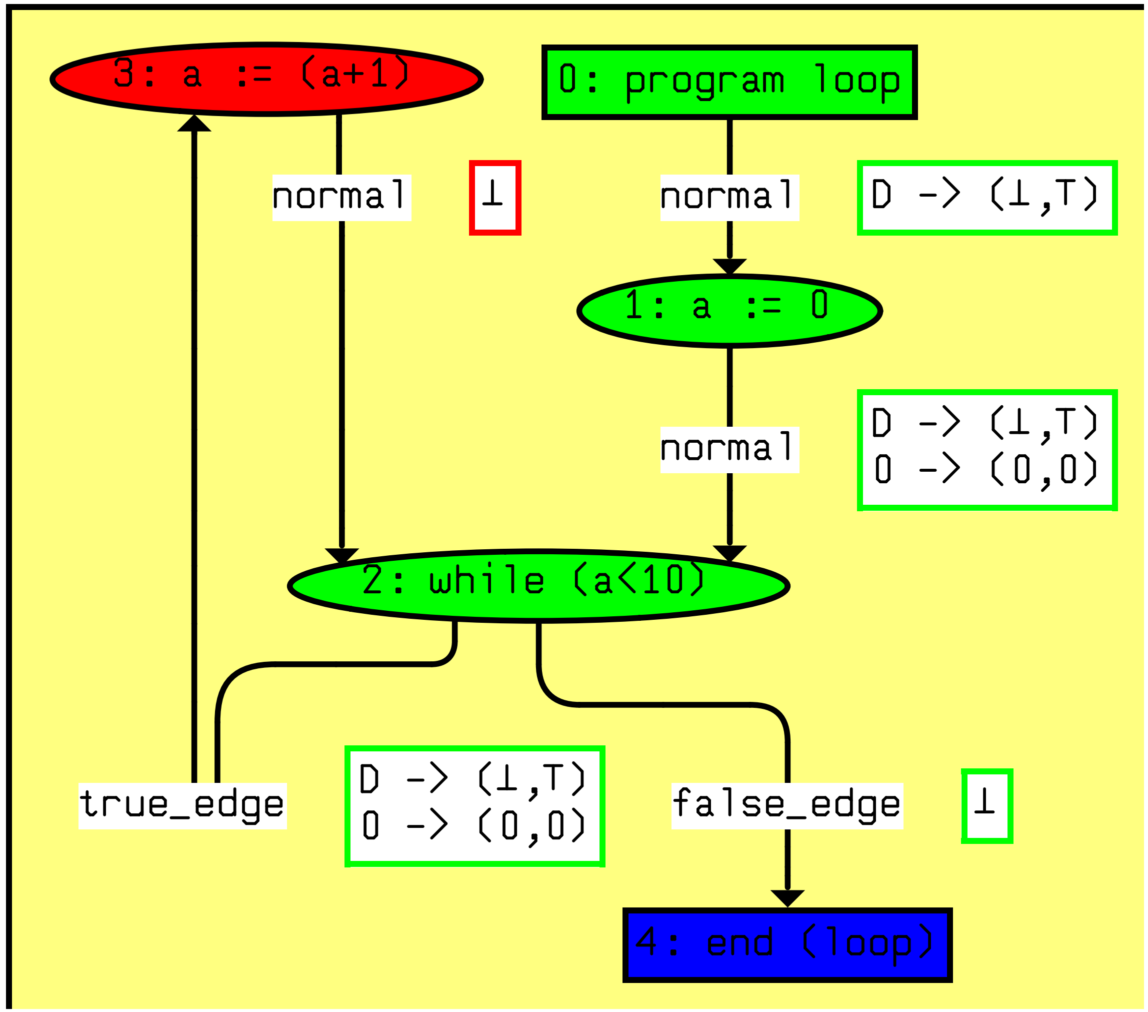
Testing the Analysis



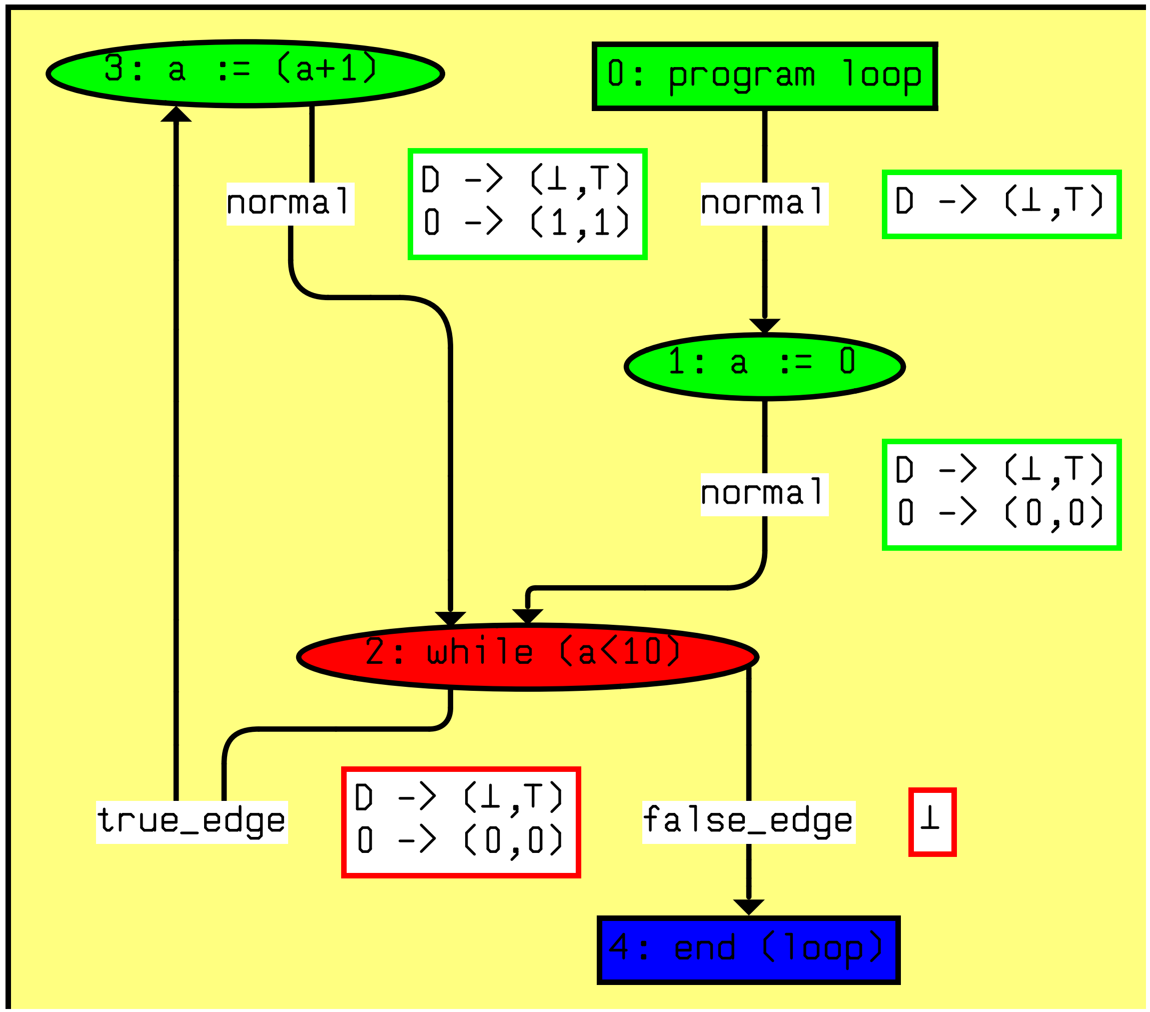
Testing the Analysis



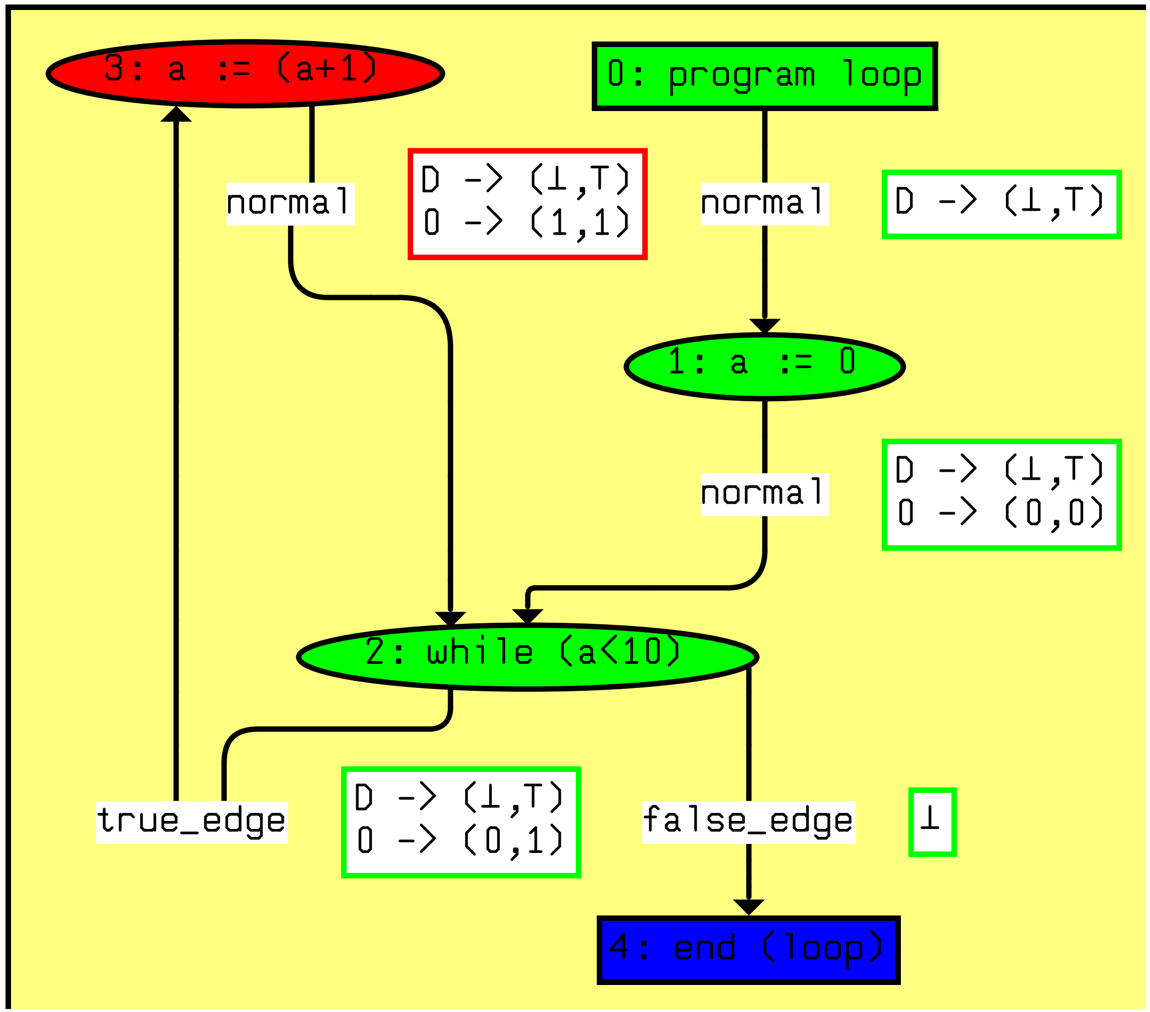
Testing the Analysis



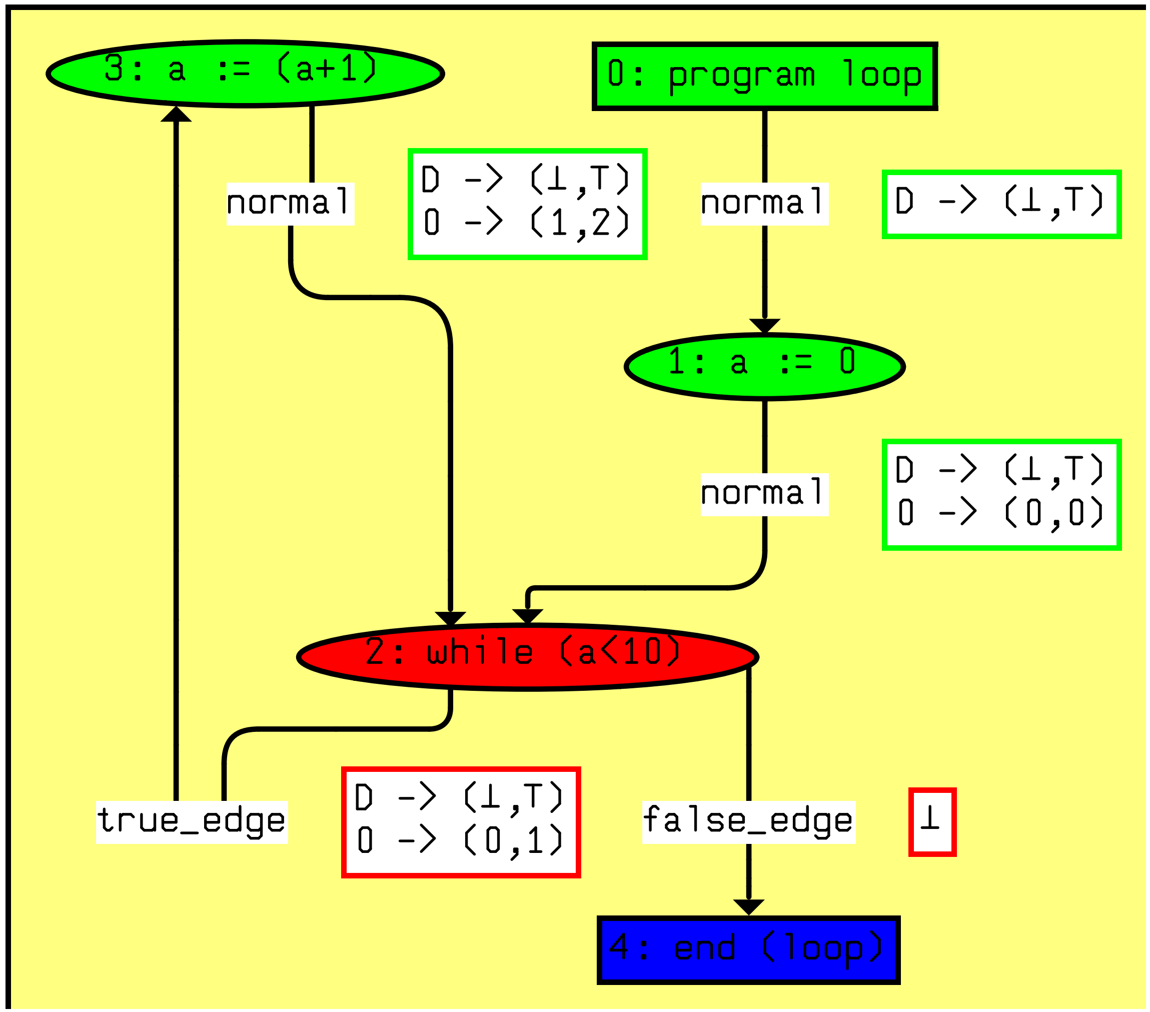
Testing the Analysis



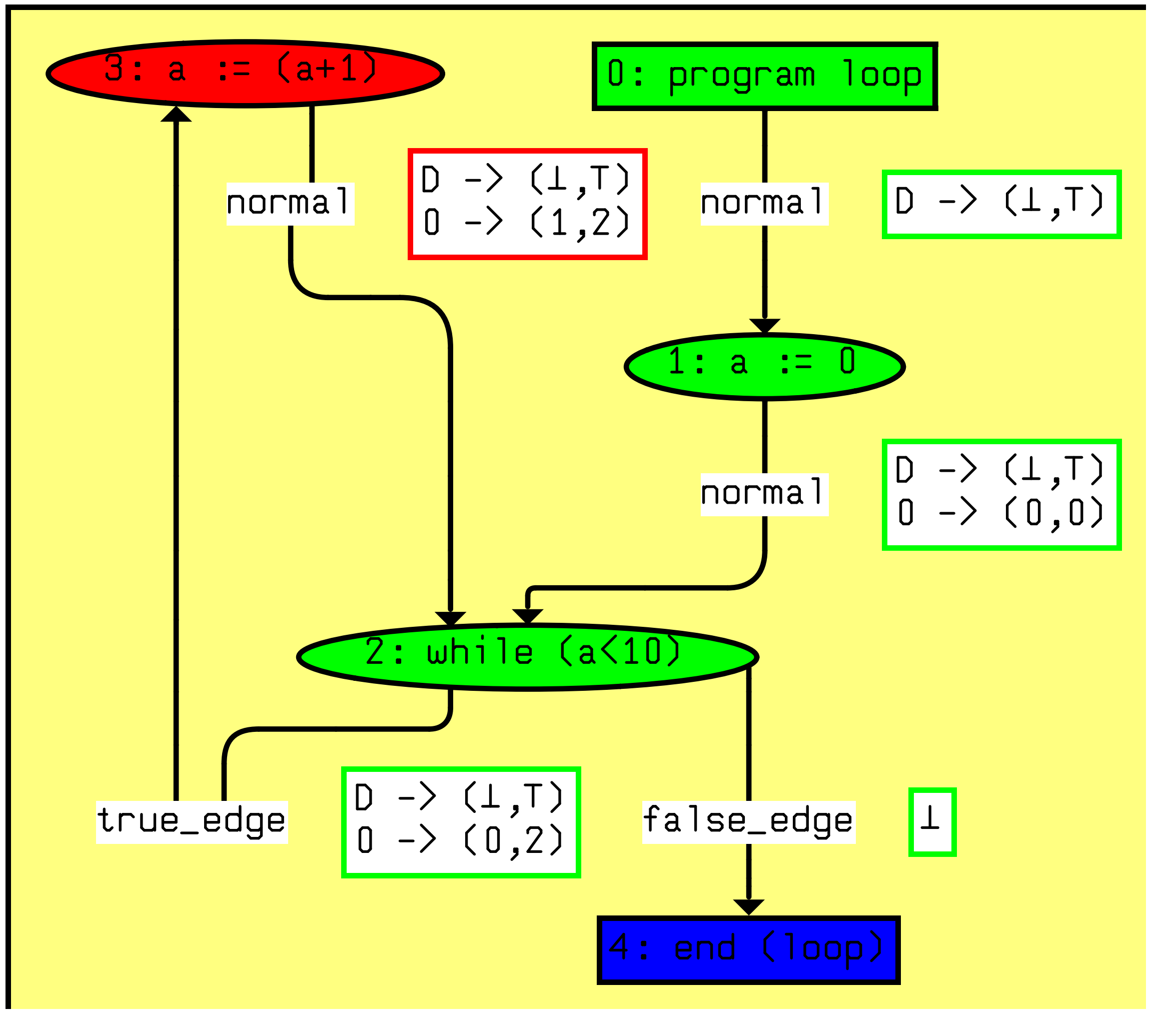
Testing the Analysis



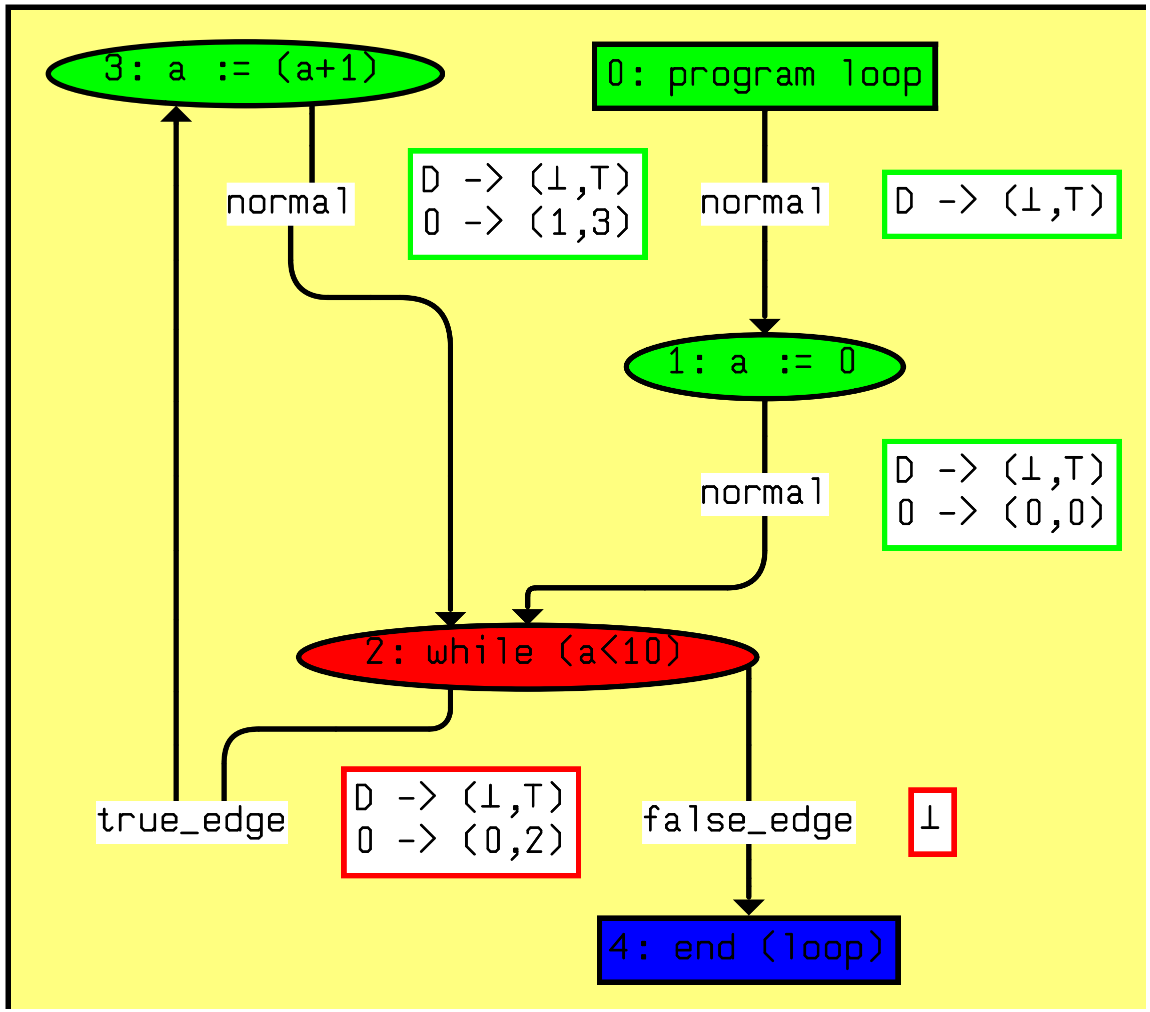
Testing the Analysis



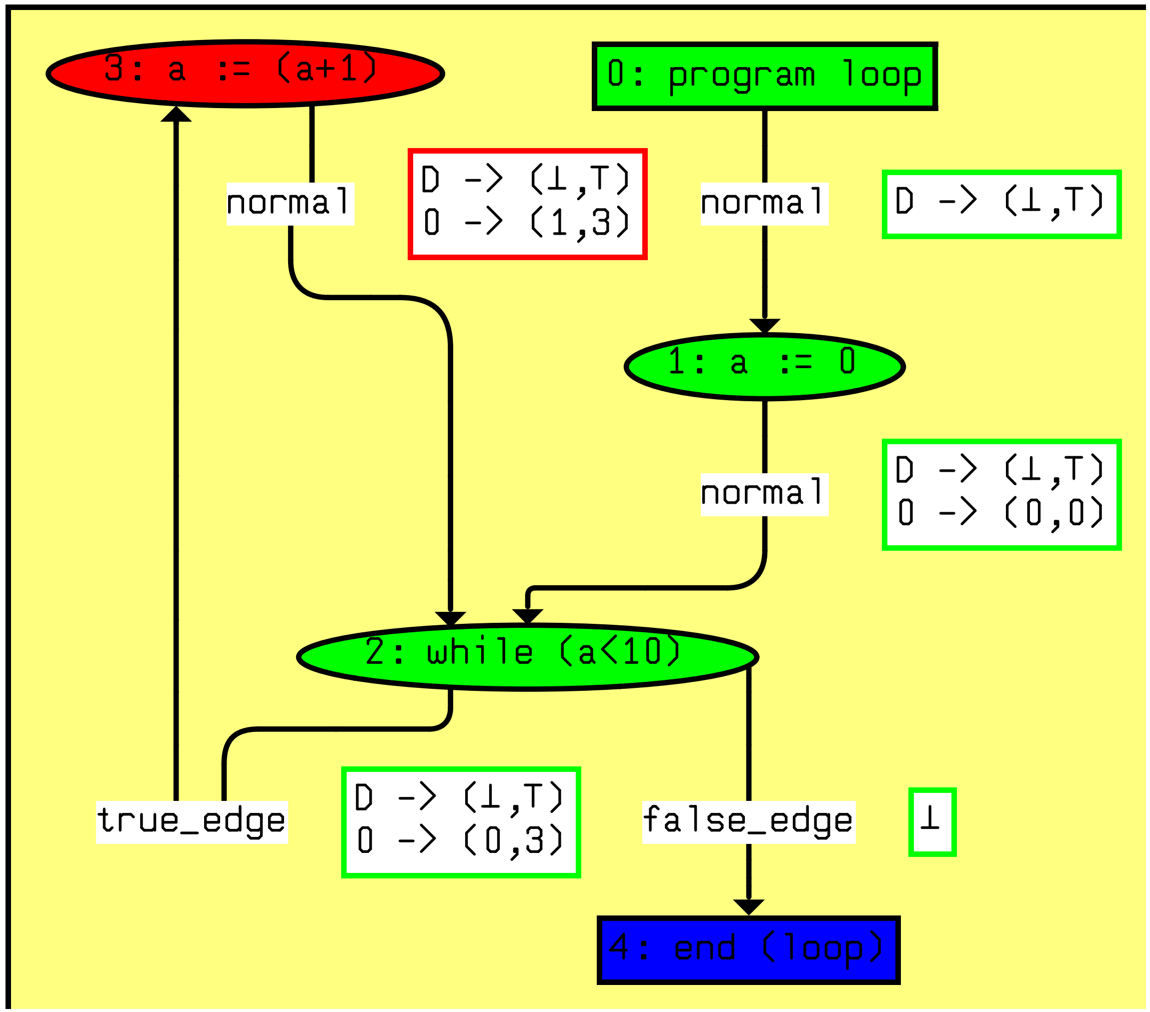
Testing the Analysis



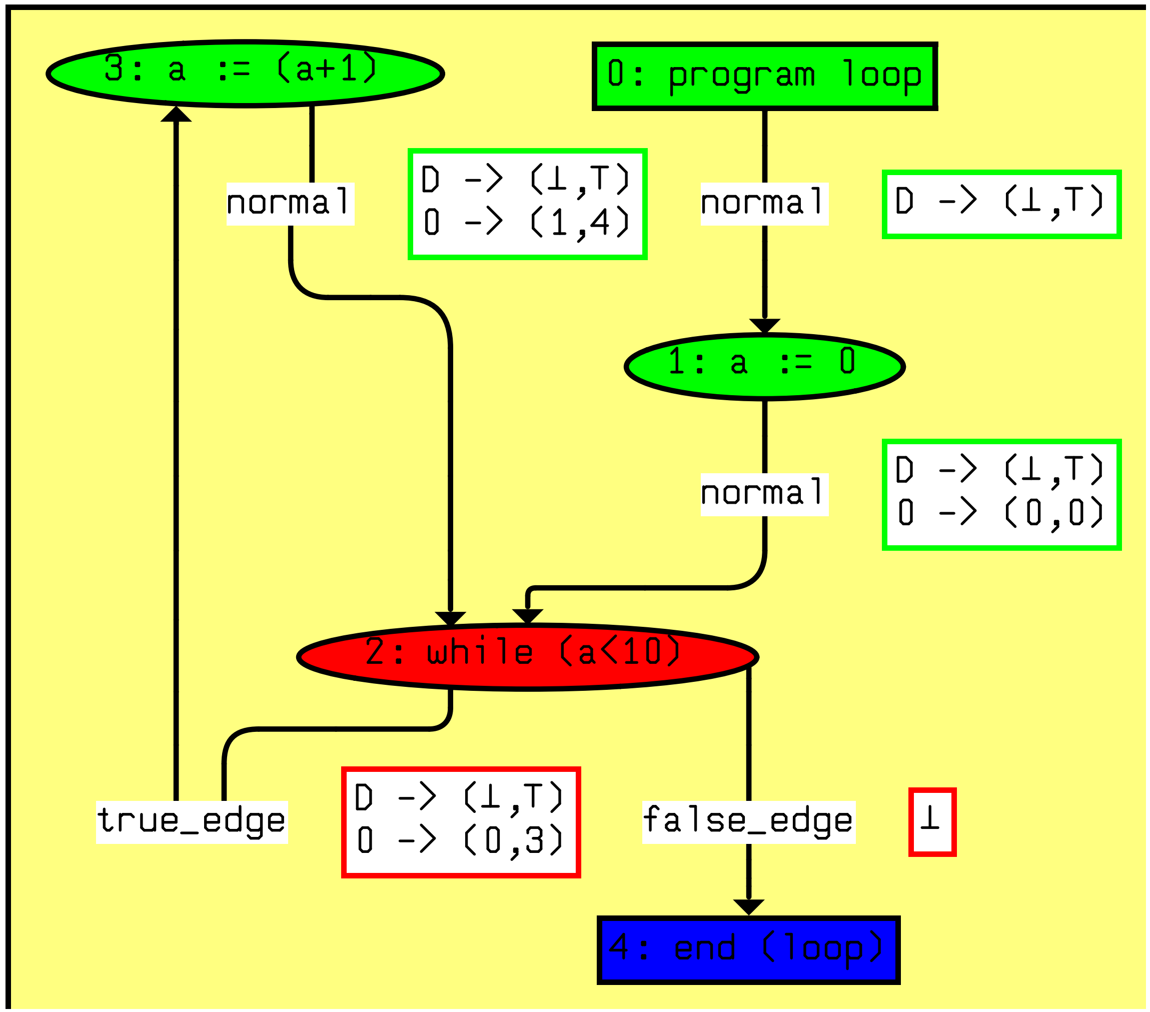
Testing the Analysis



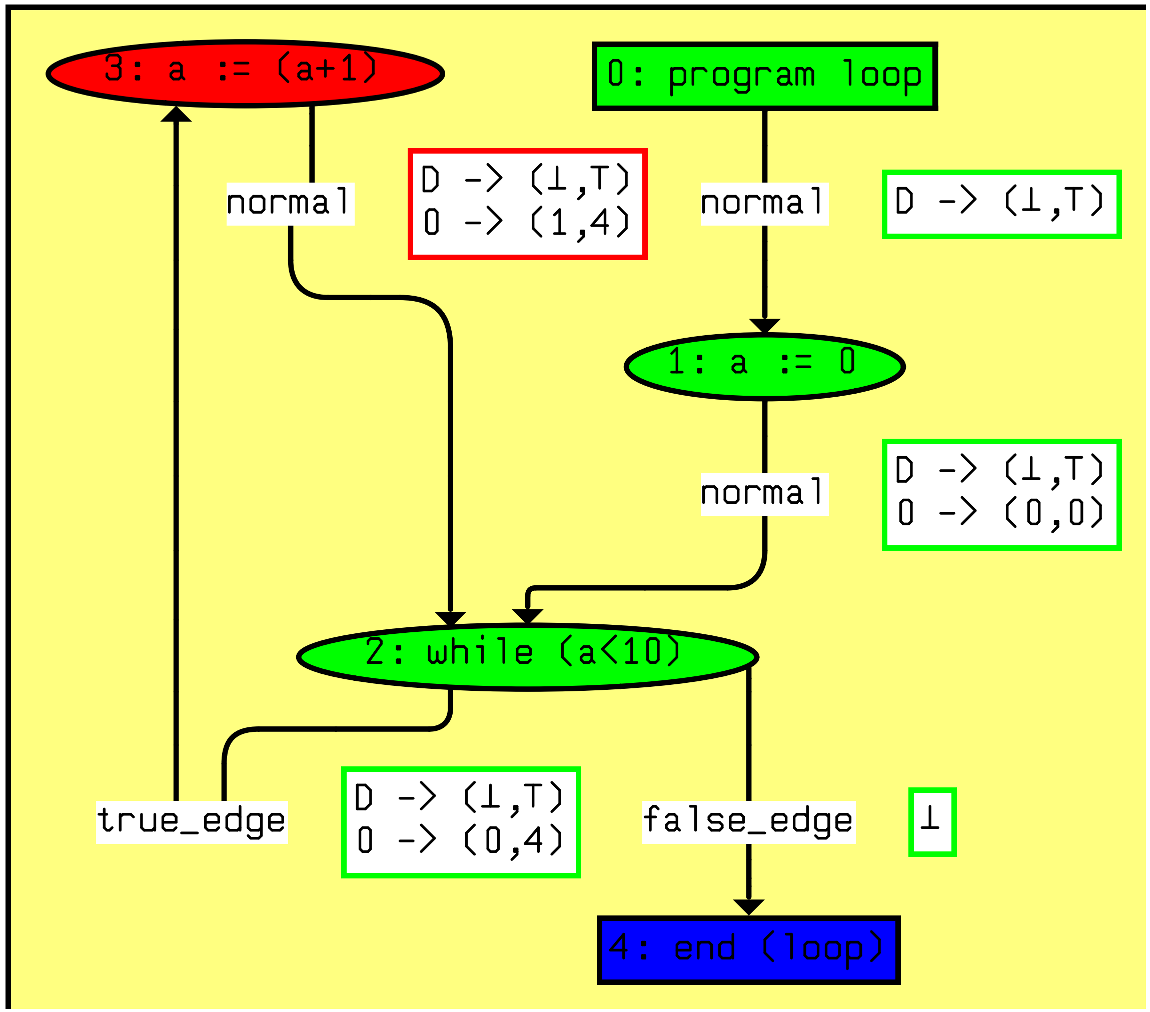
Testing the Analysis



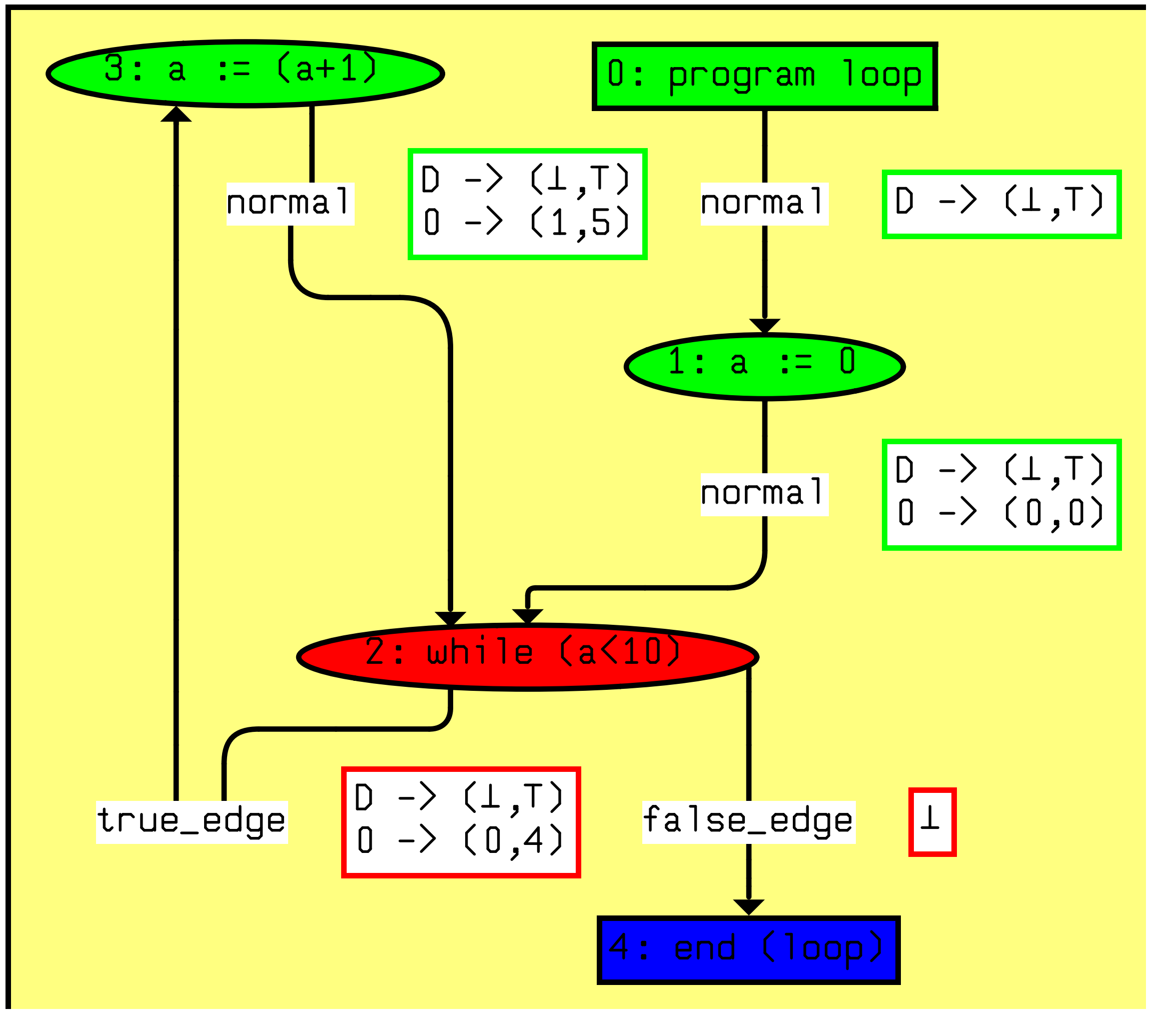
Testing the Analysis



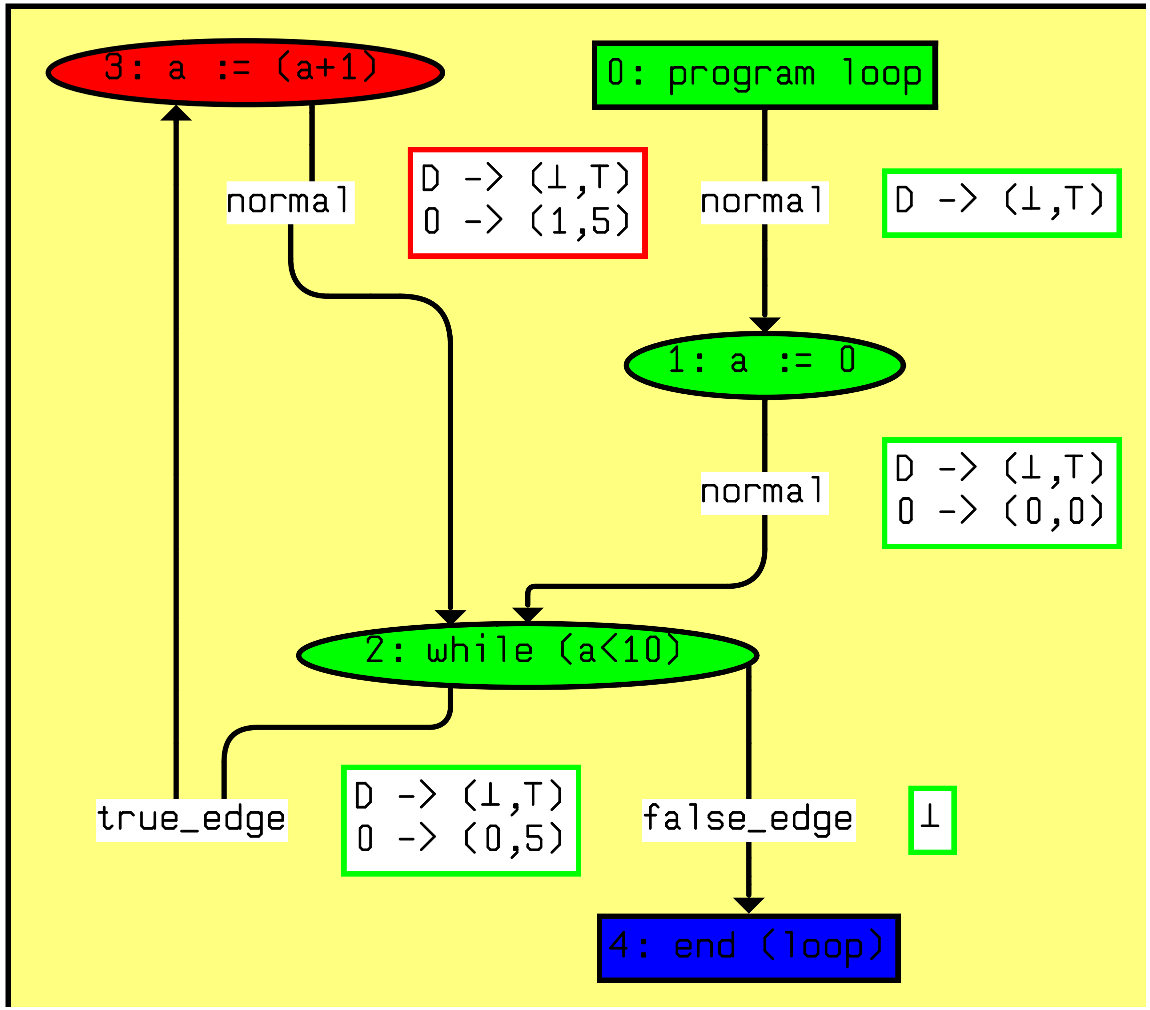
Testing the Analysis



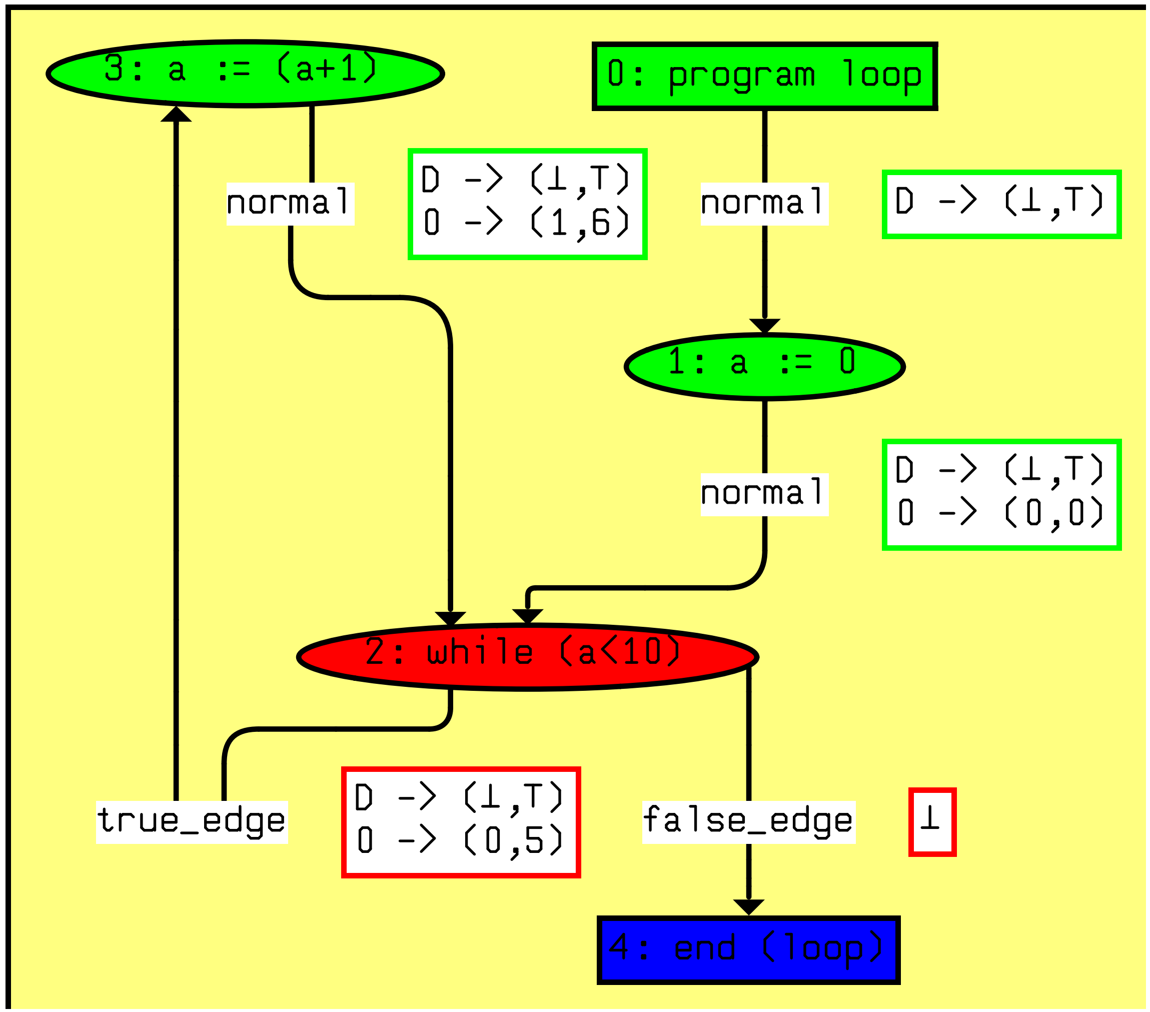
Testing the Analysis



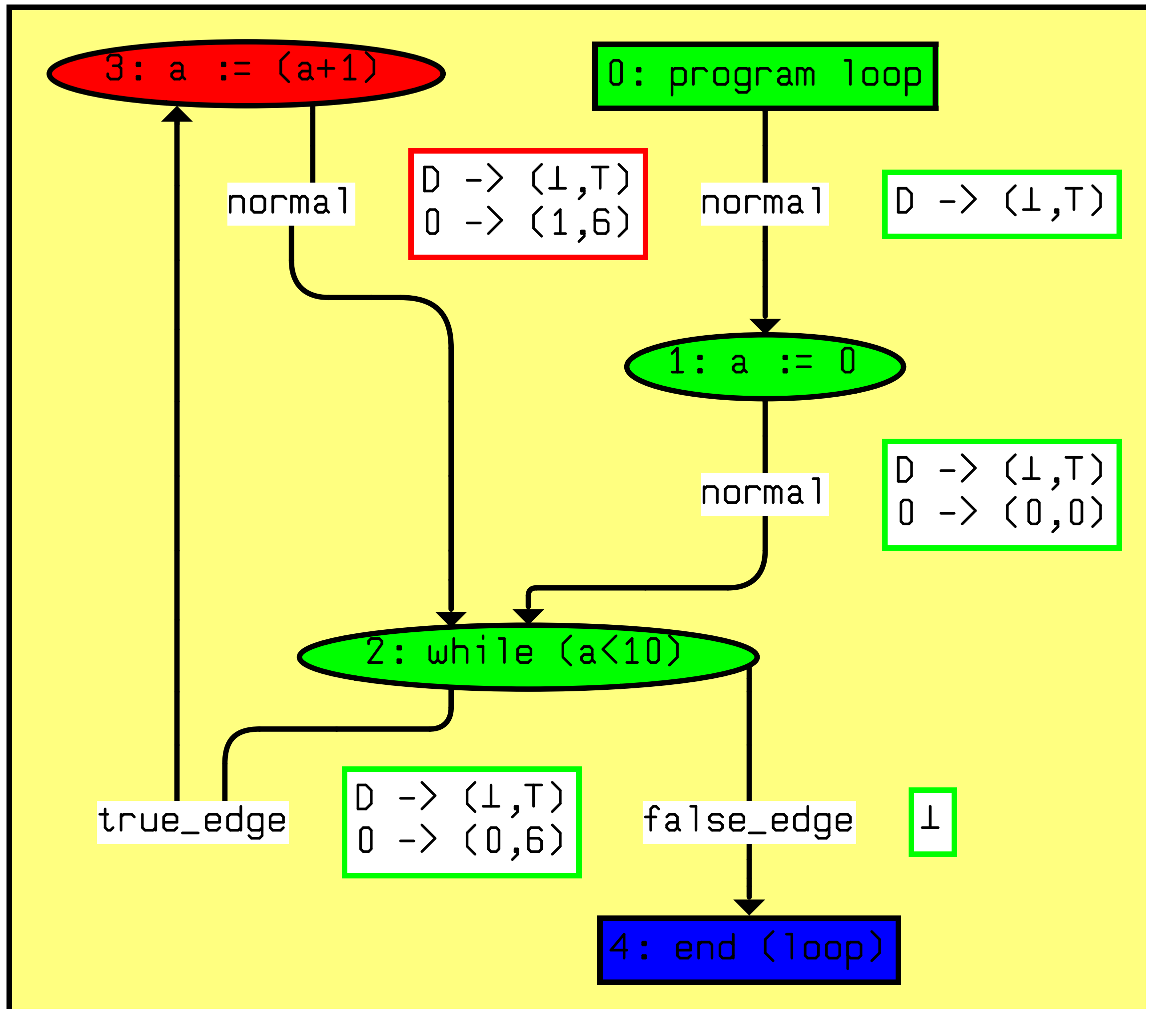
Testing the Analysis



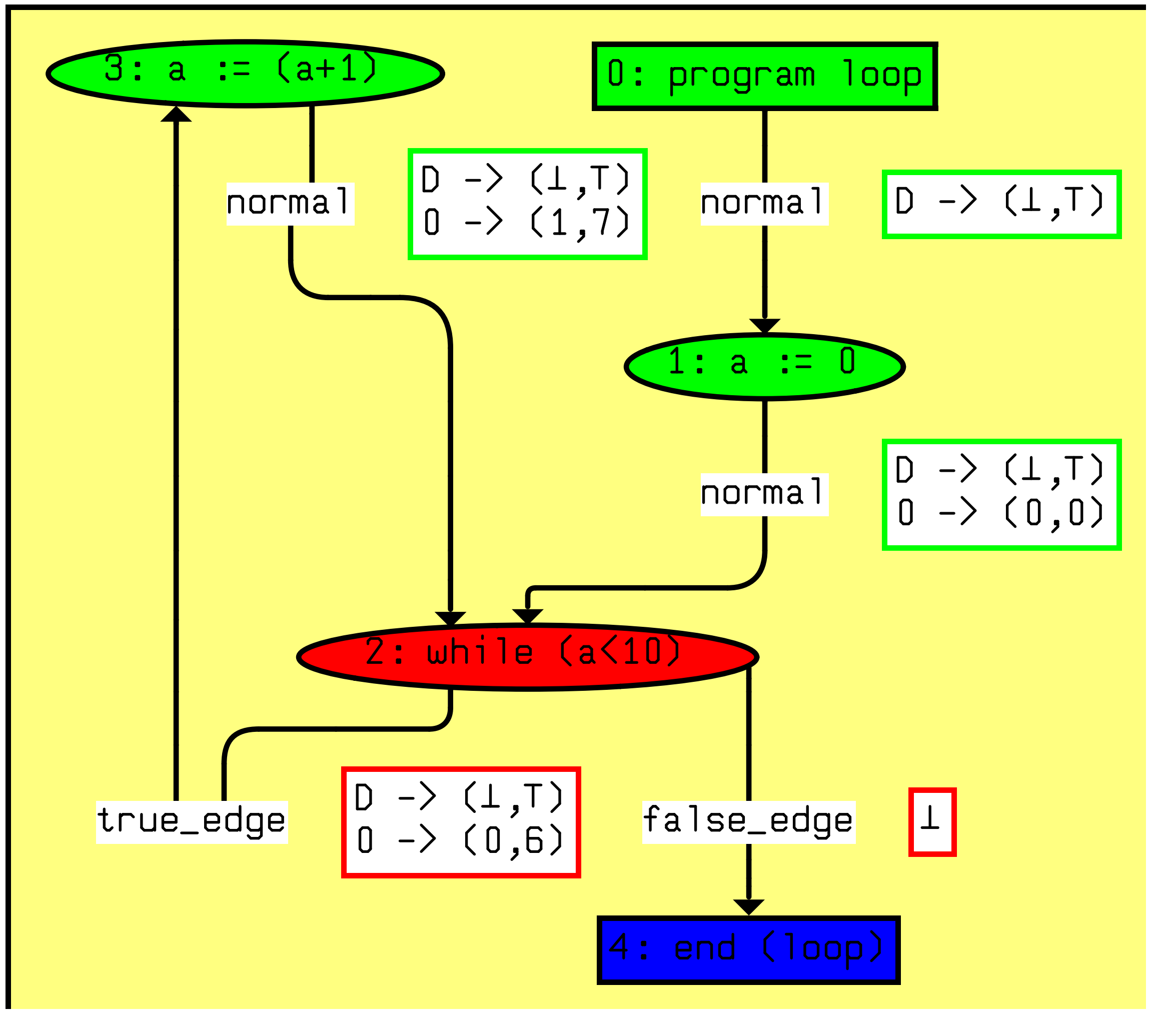
Testing the Analysis



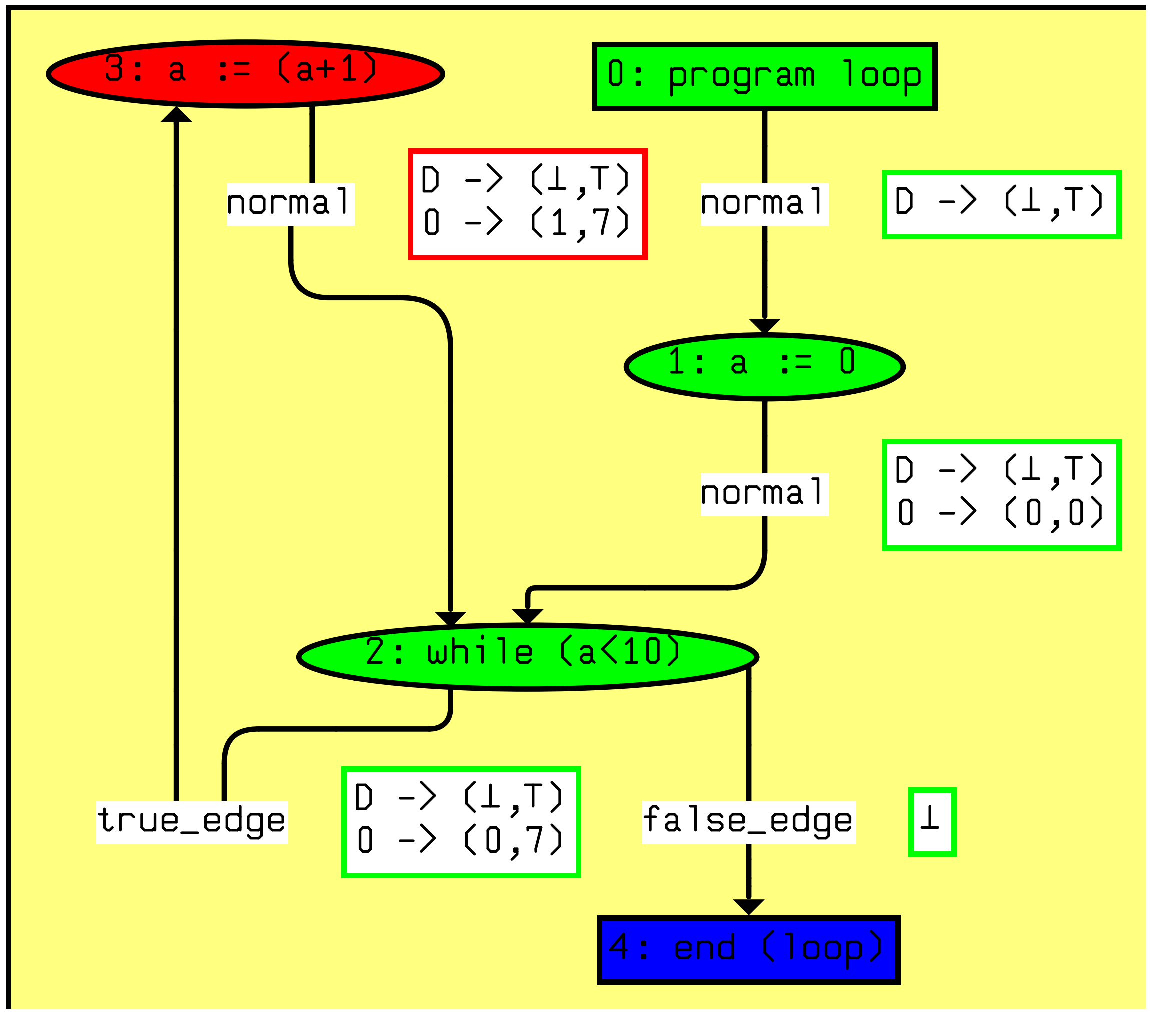
Testing the Analysis



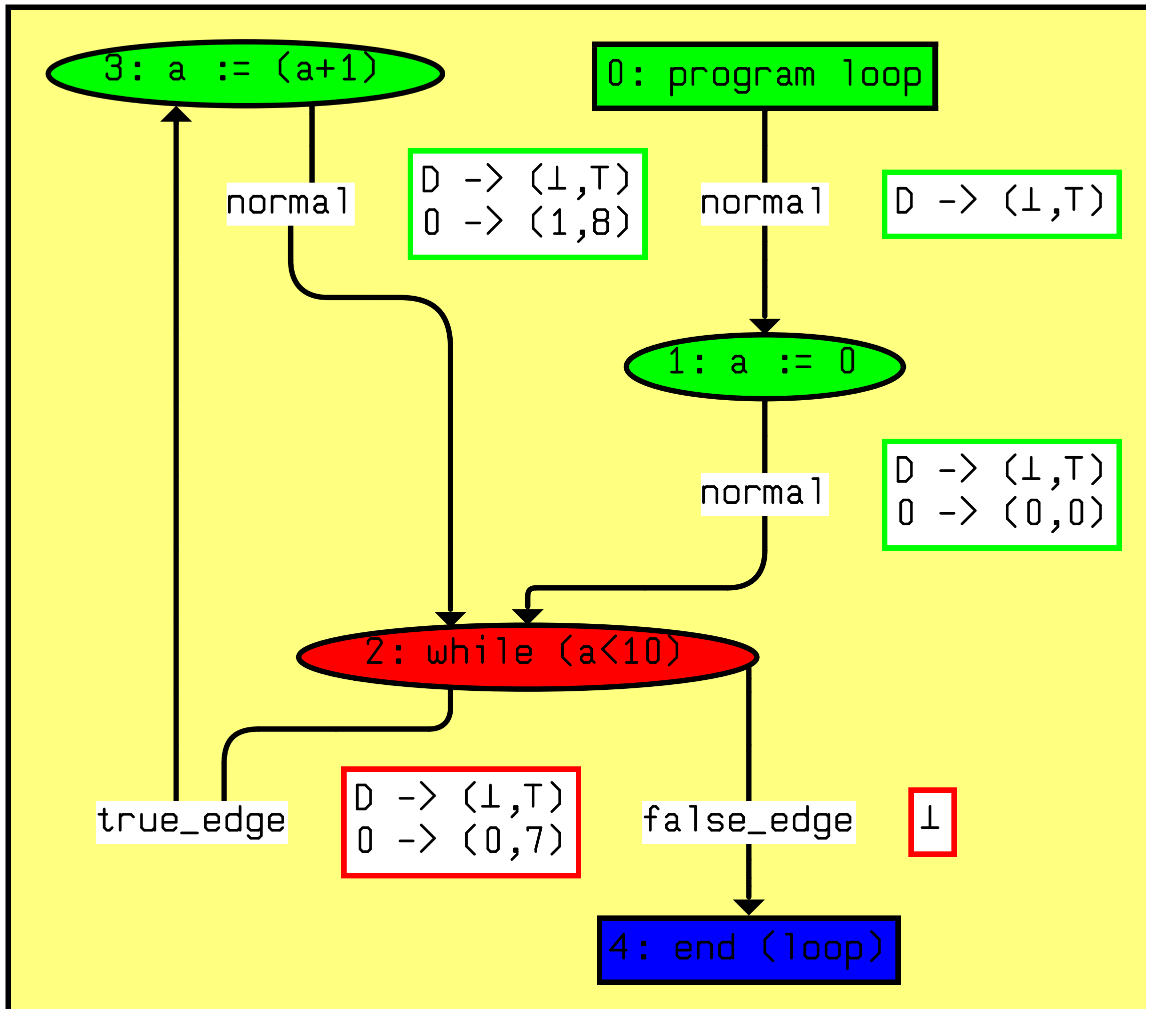
Testing the Analysis



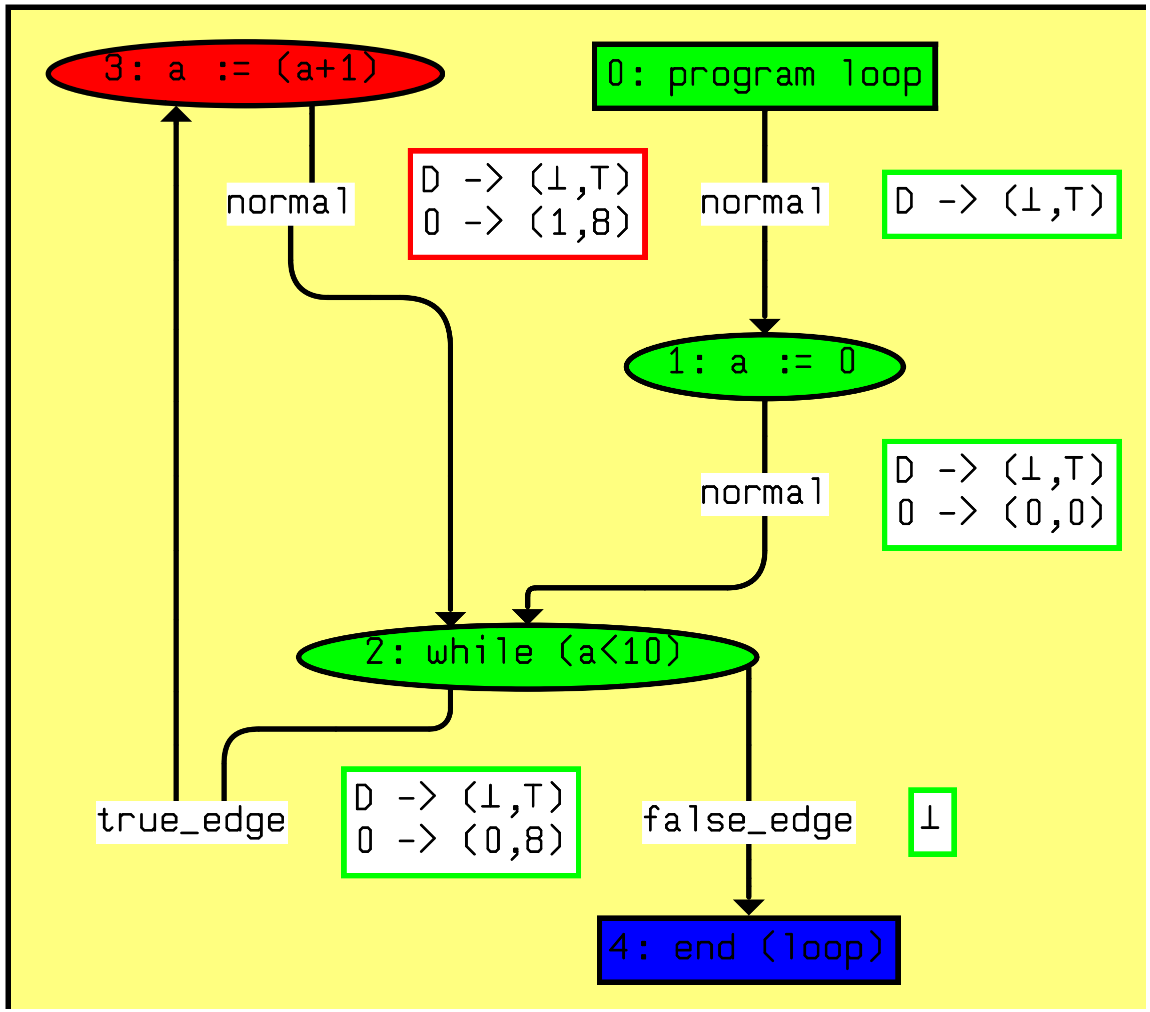
Testing the Analysis



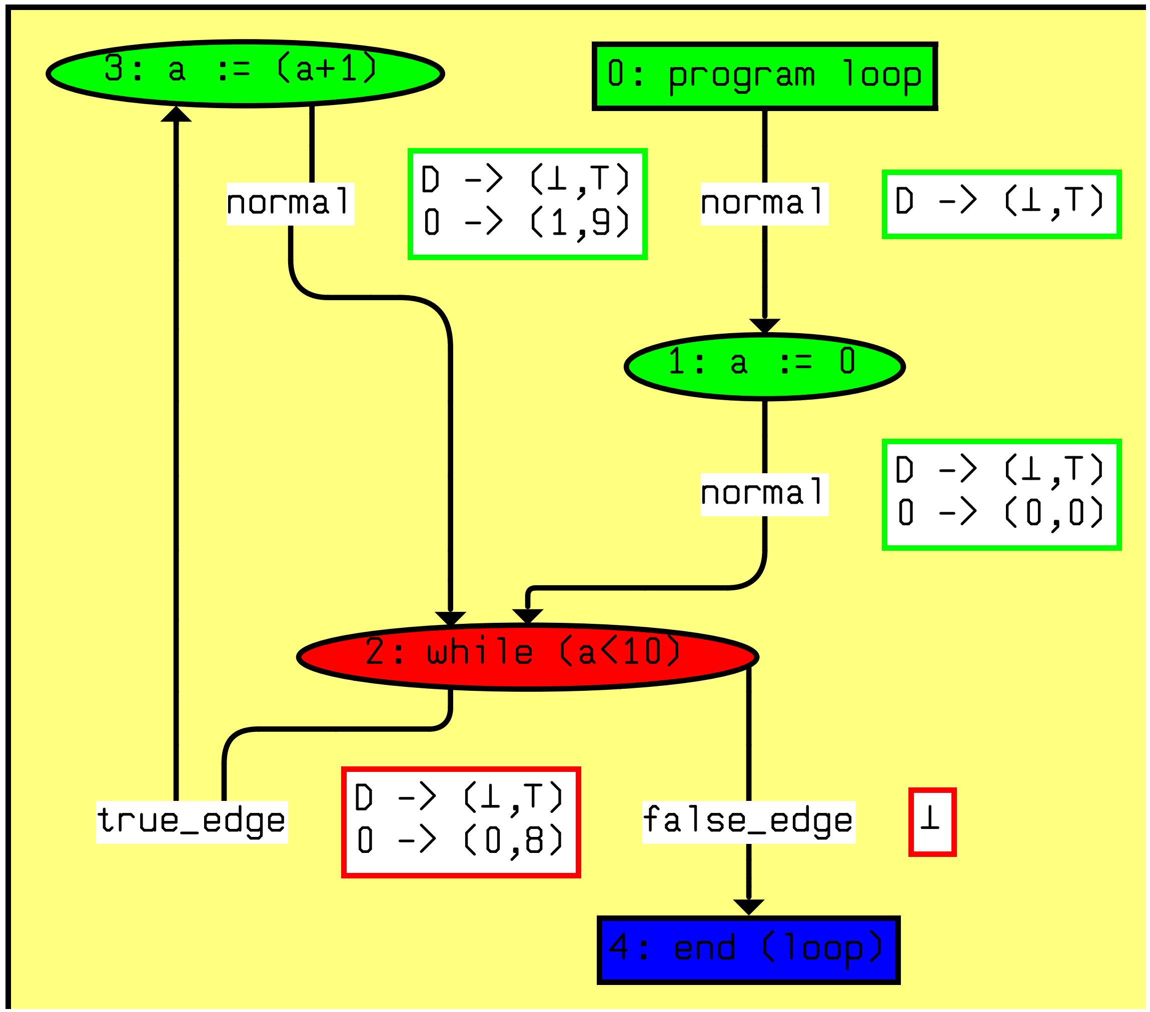
Testing the Analysis



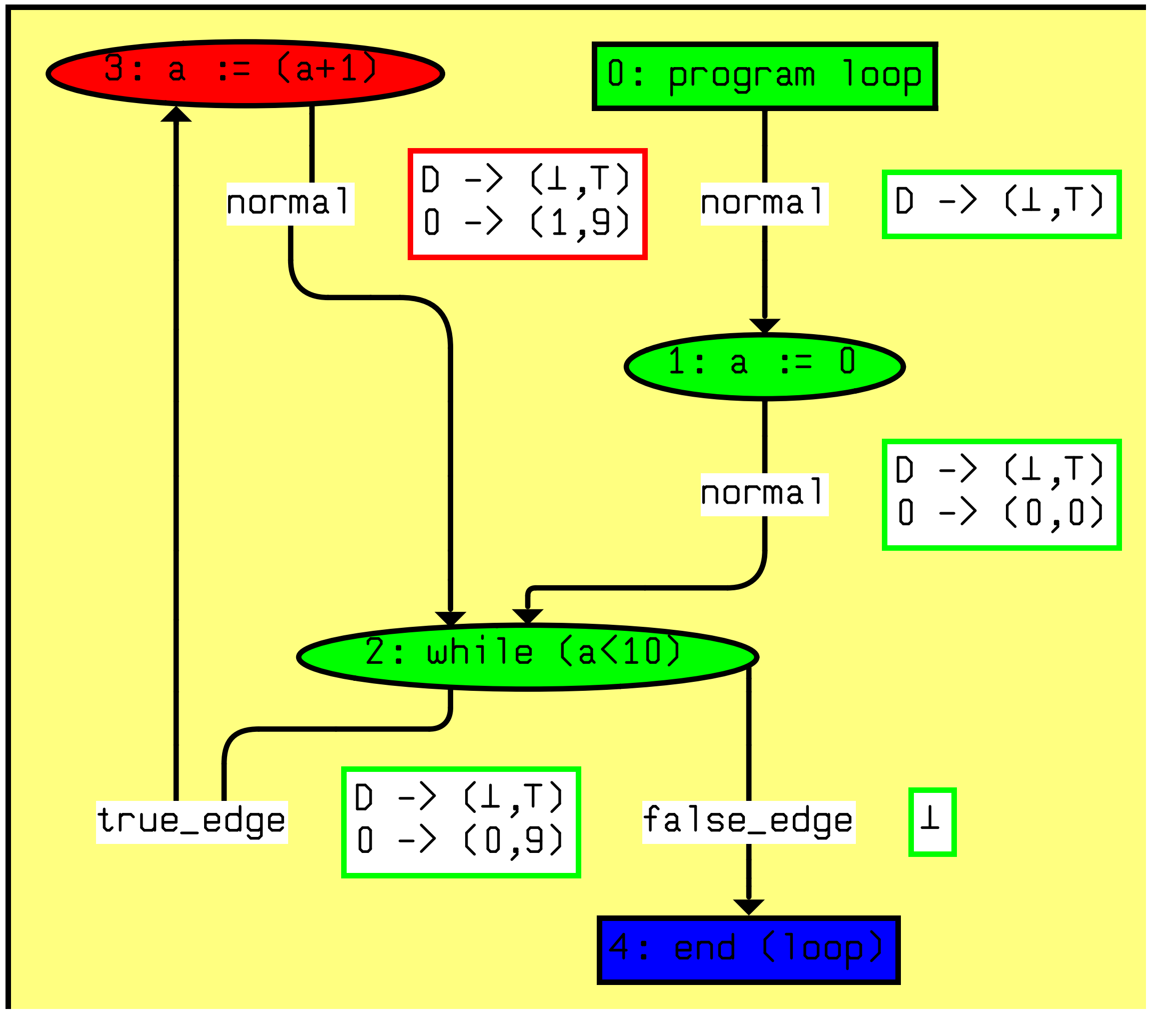
Testing the Analysis



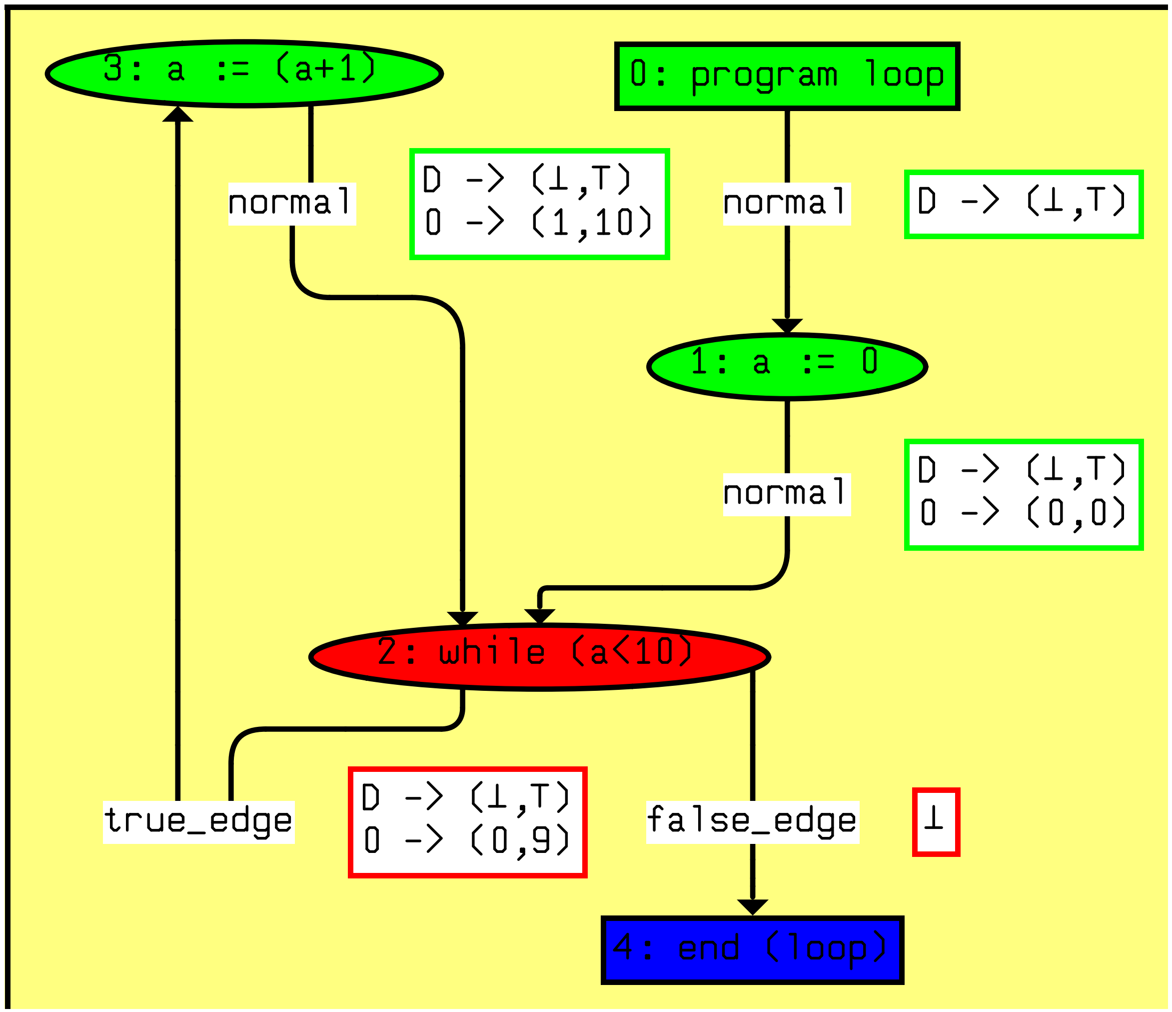
Testing the Analysis



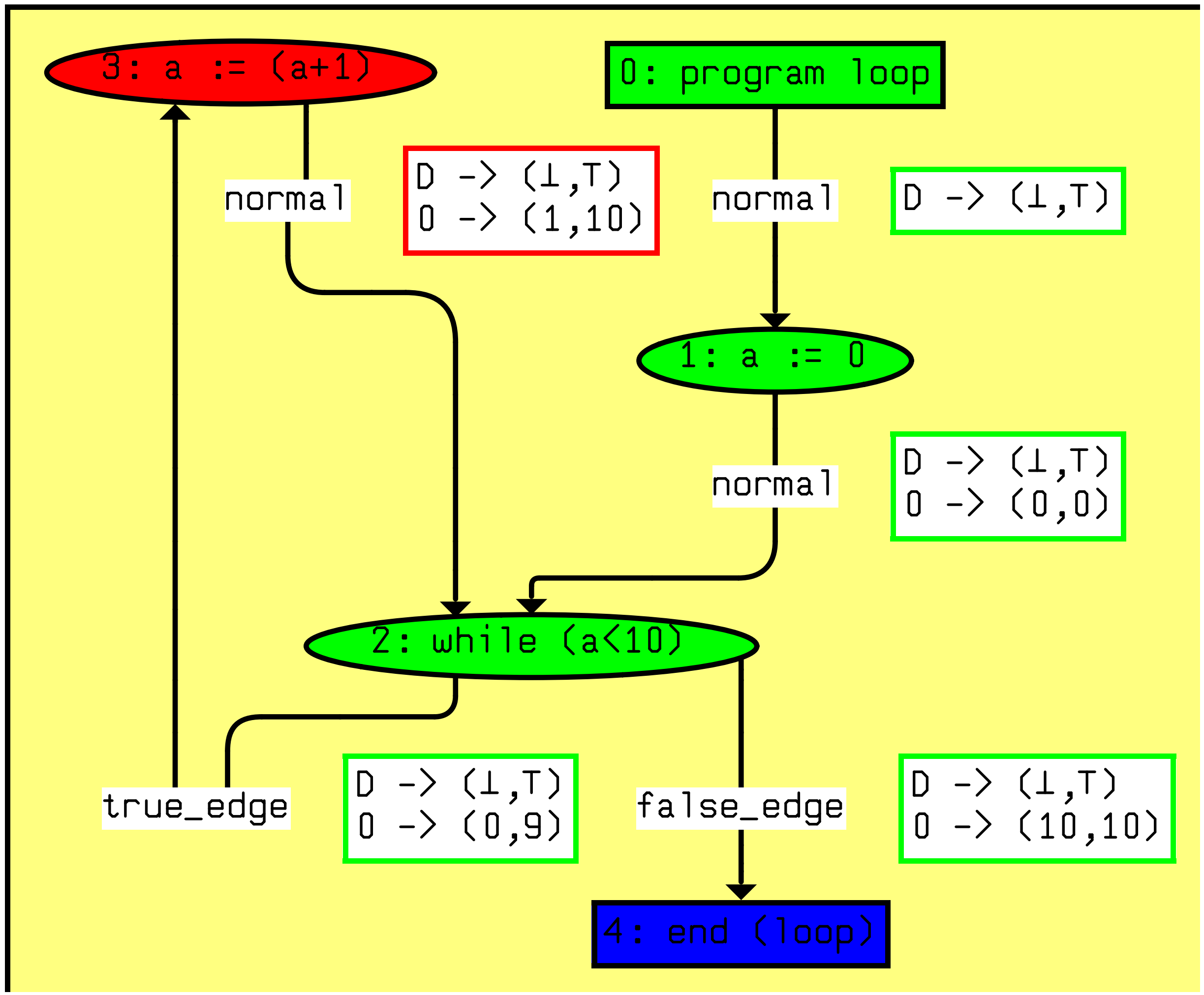
Testing the Analysis



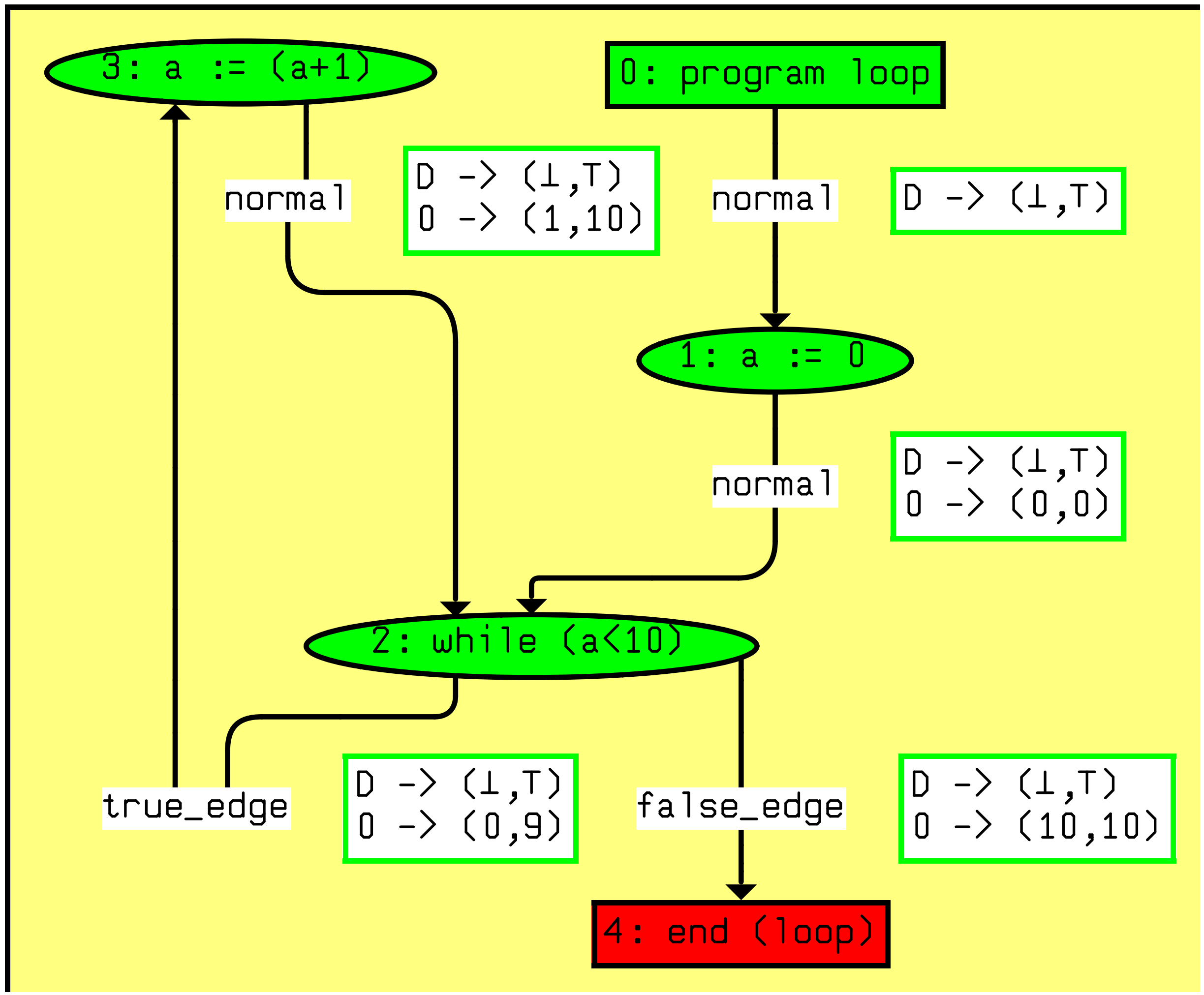
Testing the Analysis



Testing the Analysis



Testing the Analysis



Insufficiency

This will not do

- You might as well run the program. We were lucky the analysis terminated.
- We need further approximation. . . But that's not possible!
  - For each program there is a finite domain.
  - But no single finite domain will do for all programs.
  - Can we find the suitable domain by some form of textual analysis?

Textual analysis?

- Cousot brings the following examples
  - McCarty's 91-function (Bourdoncle)
  - Rational congruence analysis (Granger)
  - Linear inequality analysis (Cousot/Halbwachs)
- Here's the 91-function

$$f(n) = \begin{cases} n - 10 & \text{if } n > 100 \\ f(f(n+11)) & \text{otherwise} \end{cases}$$

How did they do it?

Widenings

- A *widening* is a lattice operator $\nabla : L \times L \to L$ such that
  - It's an upper bound operator
  - For all increasing chains $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$, the increasing chain defined by

$$\begin{aligned} y_0 &= x_0 \\ y_{i+1} &= y_i \nabla x_{i+1} \end{aligned}$$

is not strictly increasing.

Iterating with widenings

- The upward iteration sequence with widening

$$X_0 = \bot$$

$$X_{i+1} = \begin{cases} X_i & \text{if } F(X_i) \sqsubseteq X_i \\ X_i \nabla F(X_i) & \text{otherwise} \end{cases}$$

stabilizes to a safe approximation of the fix-point of F.

- F is *reductive* at that point.

Why is that?

- If we reach a point in $\mathrm{Red}(F)$, then we're ok.

$$X_0 = \bot$$

$$X_{i+1} = \begin{cases} X_i & \text{if } F(X_i) \sqsubseteq X_i \\ X_i \nabla F(X_i) & \text{otherwise} \end{cases}$$

- The sequence is clearly ascending.
- If $\exists i : F(X_i) \sqsubseteq X_i$, then
  - The sequence will immediately stabilize.
  - Since $X_i \in \mathrm{Red}(F)$, we have $\mathrm{lfp}(F) \sqsubseteq X_i$.

Will we get there?

- Assume $\forall i : X_i \sqsubset F(X_i)$, then

$$X_0 = \bot$$
$$X_{i+1} = X_i \nabla F(X_i)$$

- This is just the widening of the following ascending chain

$$Y_0 = \bot$$
$$Y_{i+1} = F(X_i)$$

QED! Really!

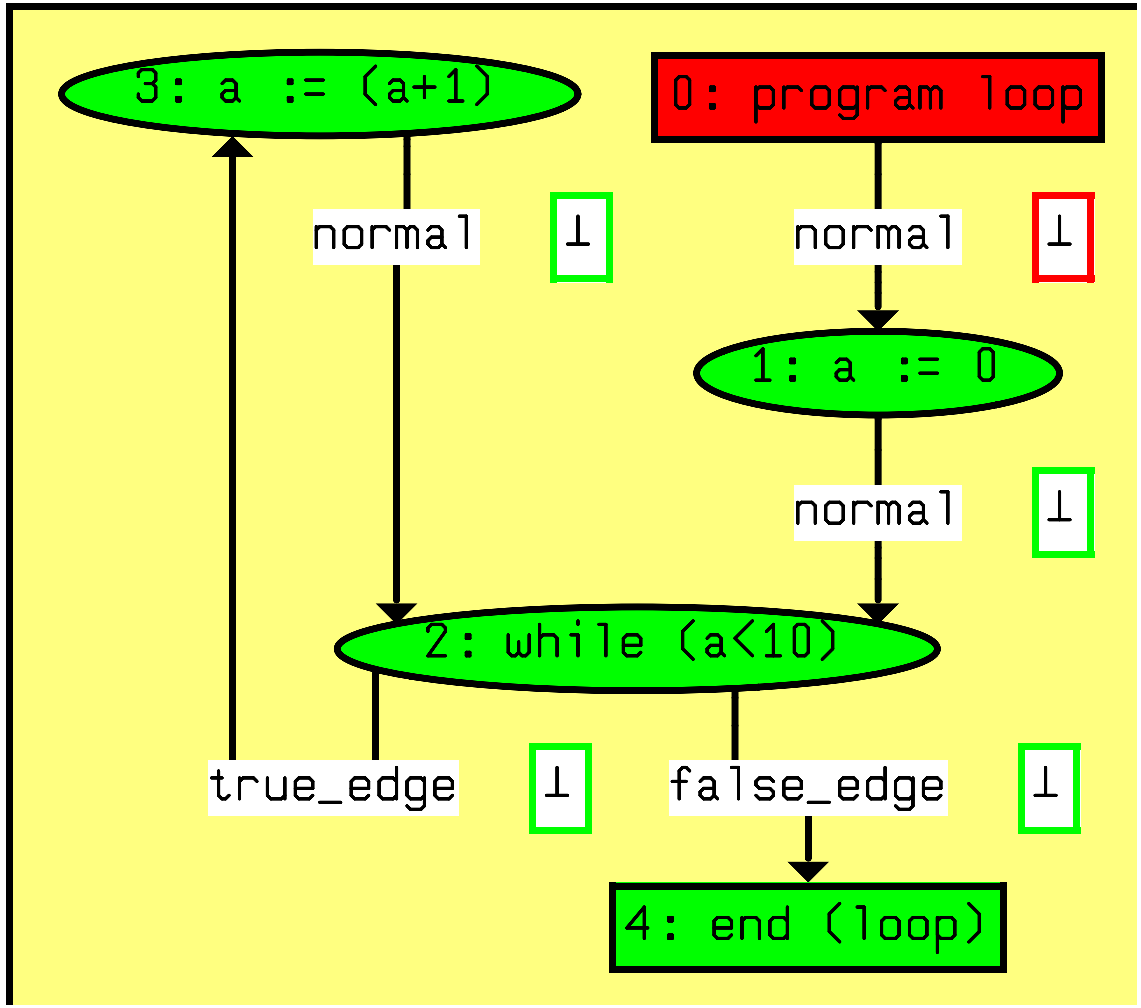
Widening for Upper bounds

- The upper bound of an interval, can be widened by

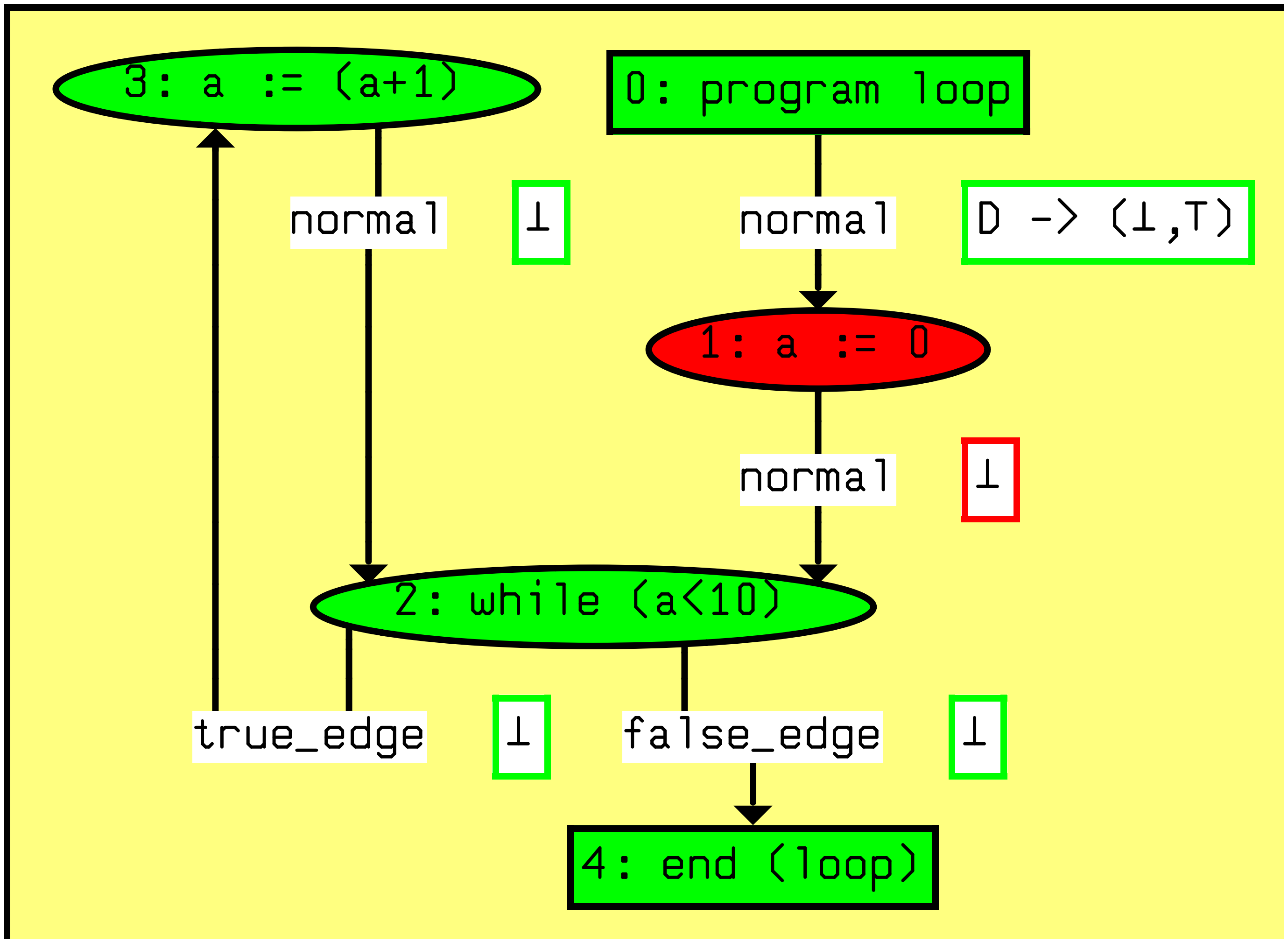
$$x \nabla y = \begin{cases} \infty & \text{if } x < y \\ x & \text{otherwise} \end{cases}$$

- It's a widening! Any chain is stablized immediately.

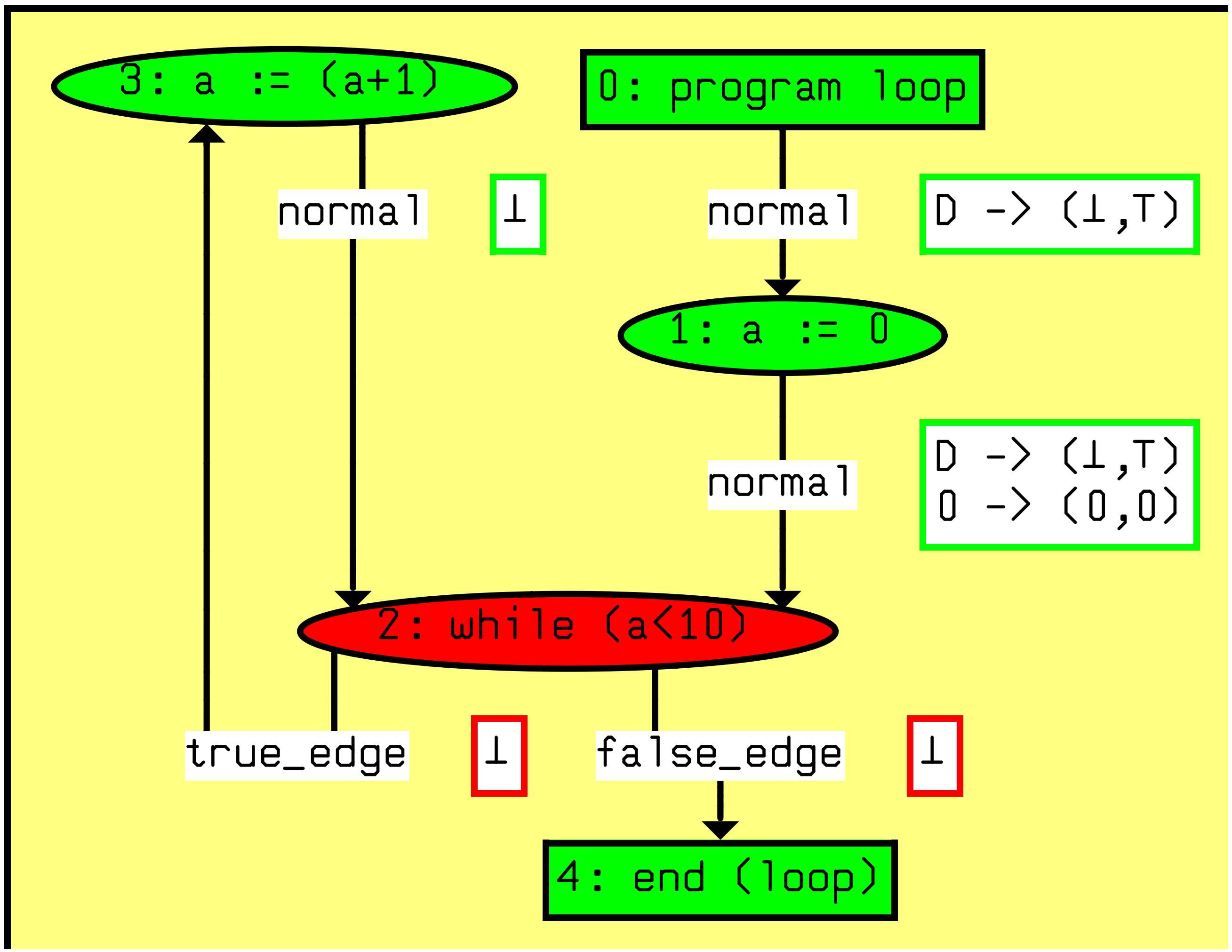
Approximating the Fixed-Point



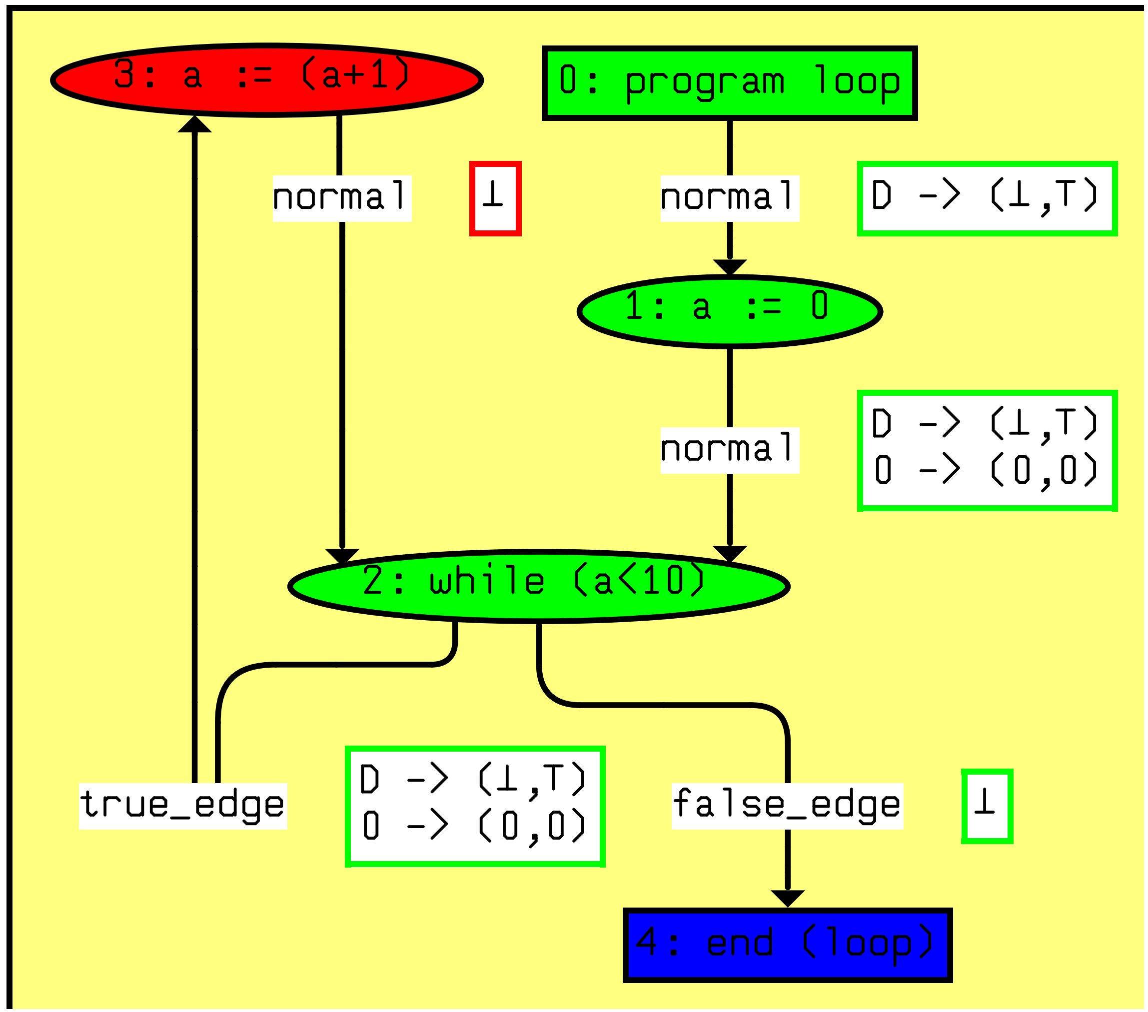
Approximating the Fixed-Point



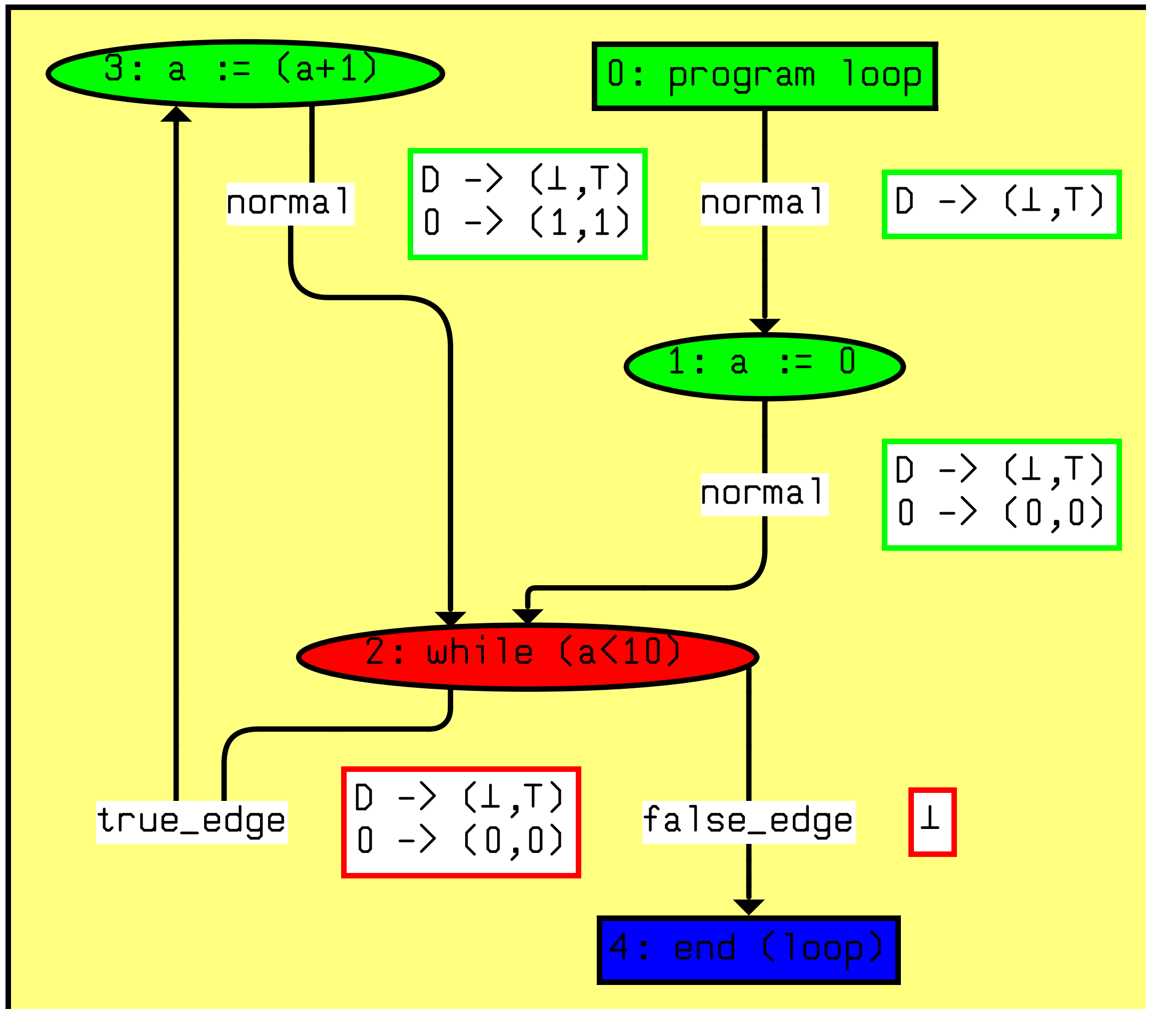
Approximating the Fixed-Point



Approximating the Fixed-Point

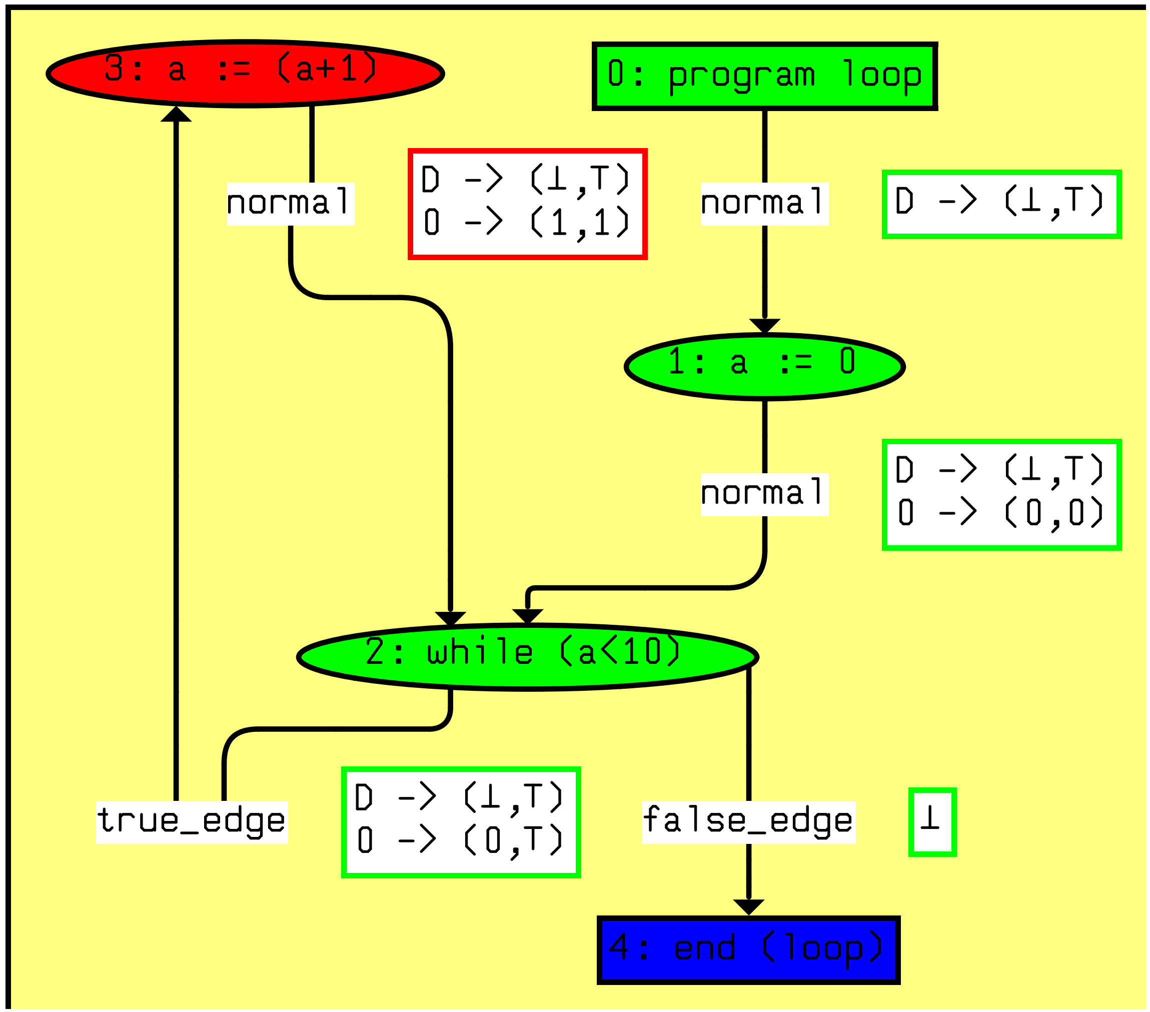


Approximating the Fixed-Point

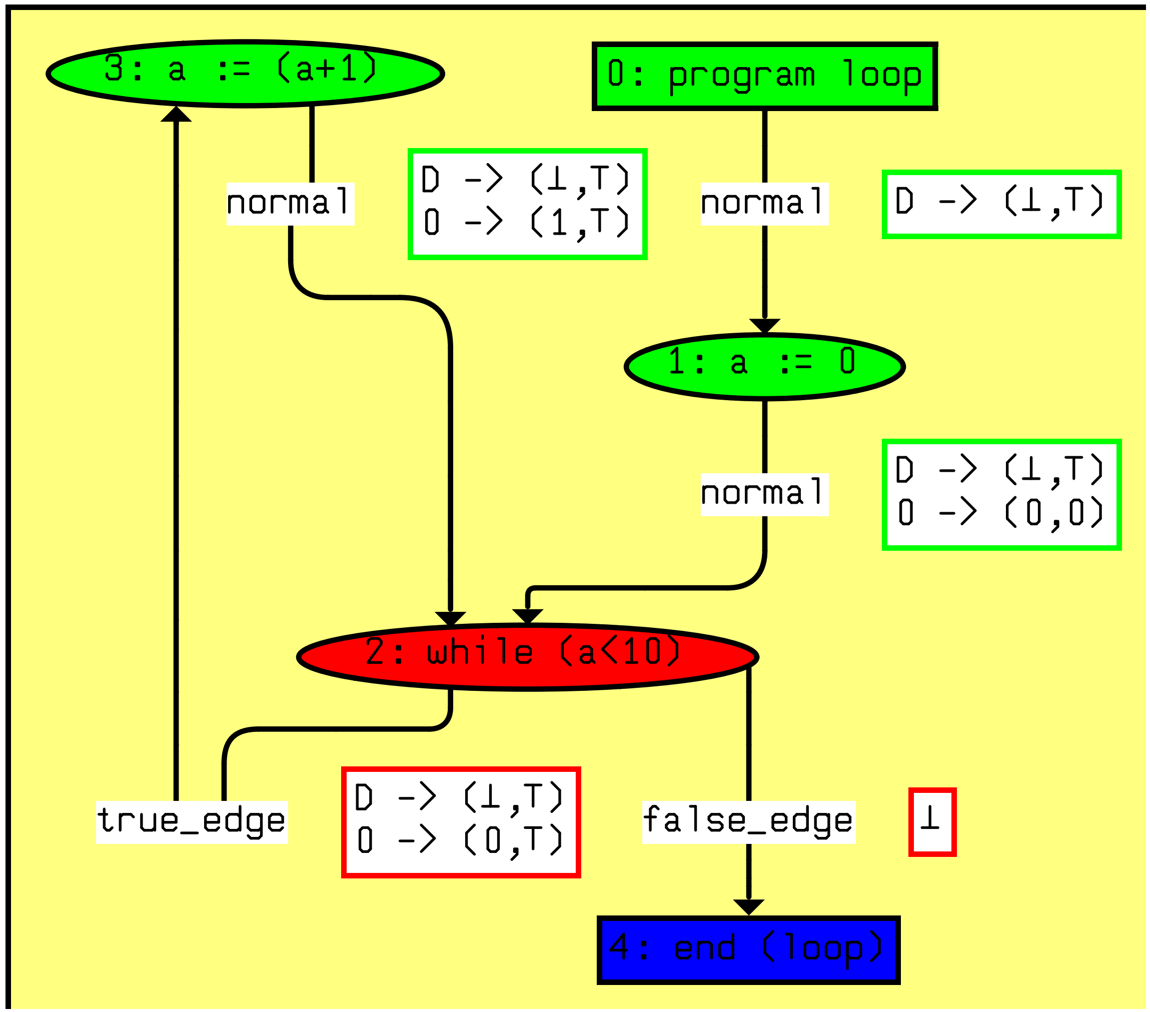


widening about to happen!

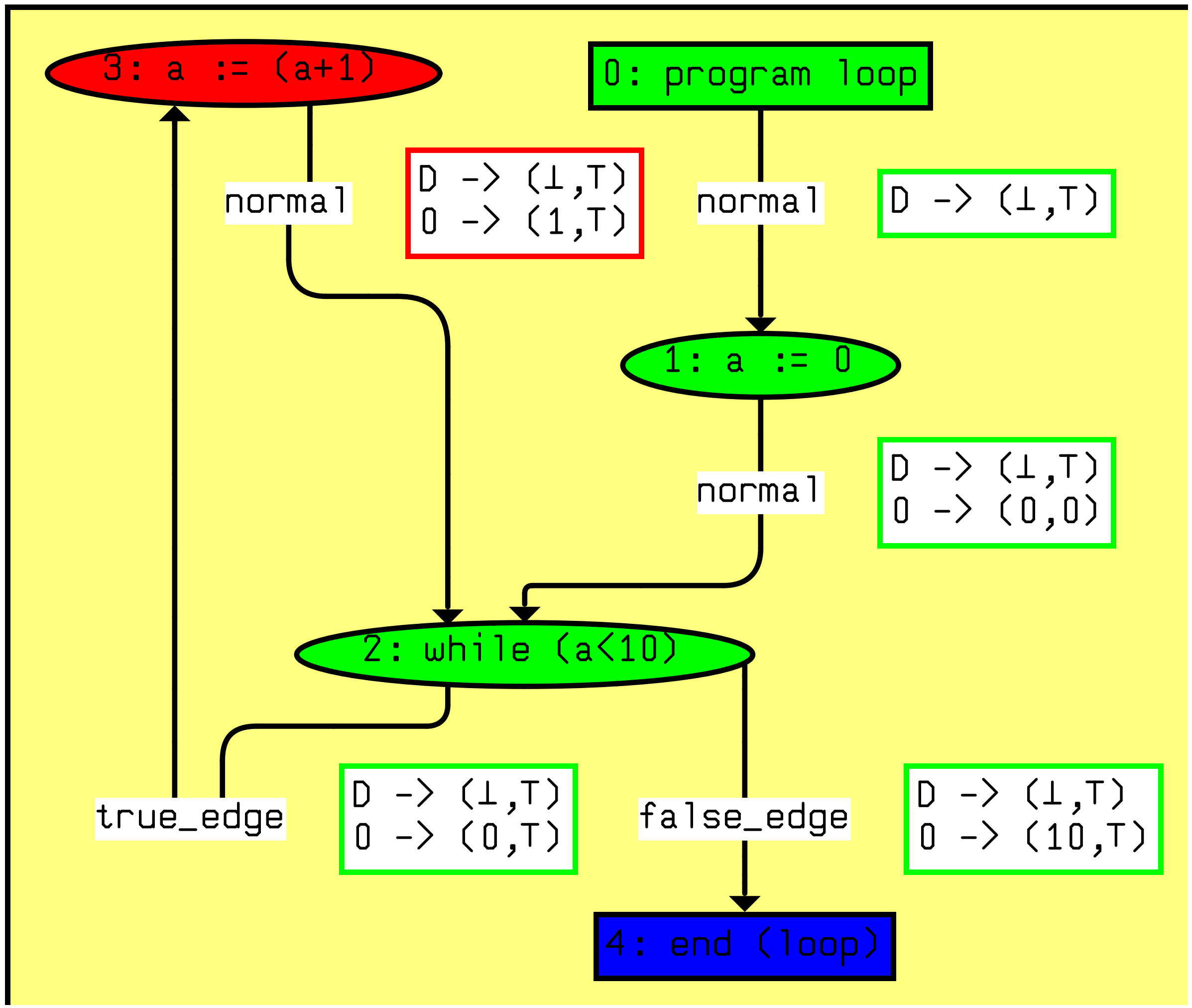
Approximating the Fixed-Point



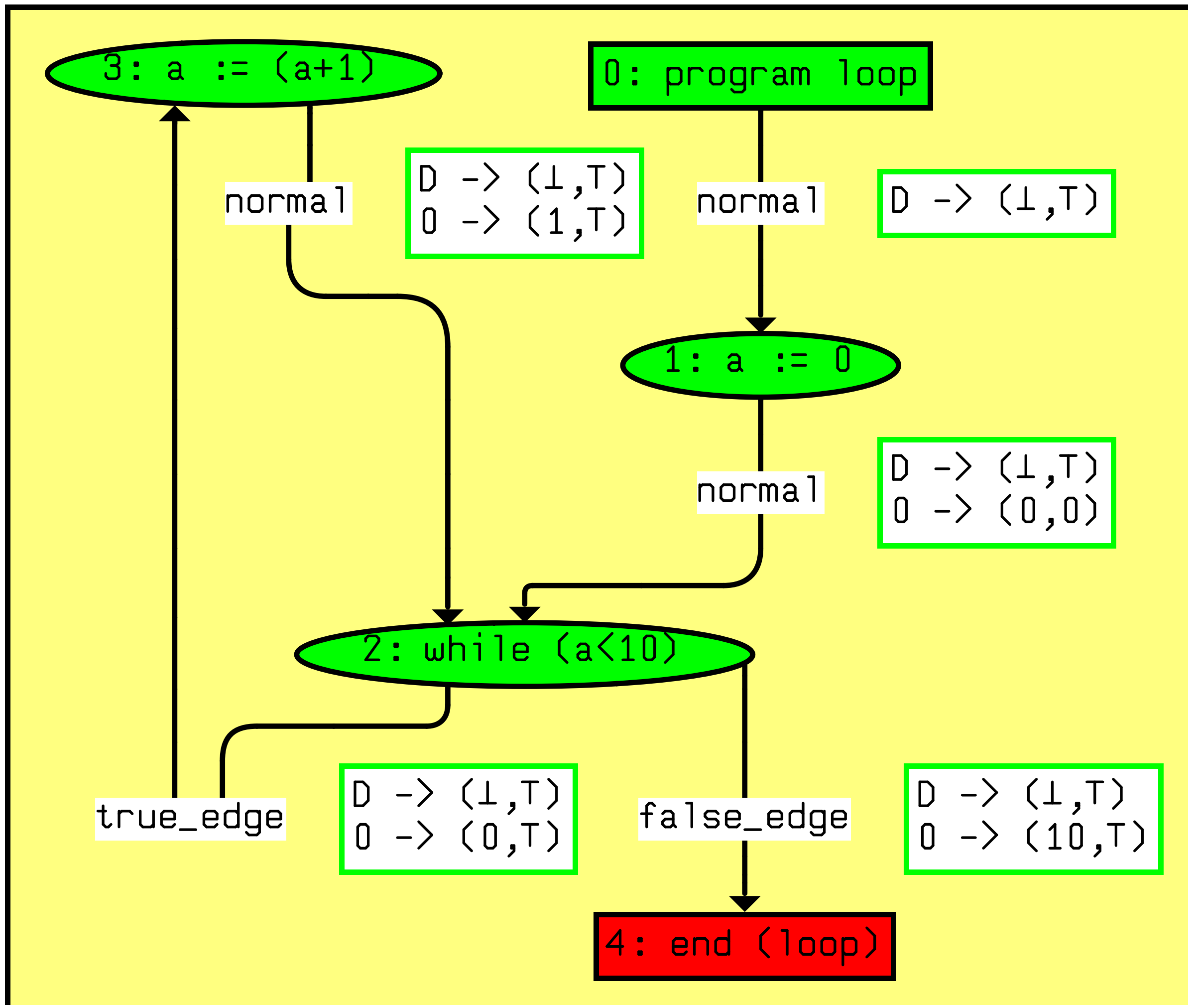
Approximating the Fixed-Point



Approximating the Fixed-Point



Approximating the Fixed-Point



Narrowings

- Since F is reductive at the point reached through the upward iteratoin, we can fine-tune the result.
- Downward sequence may also not reach the fix-point, so we might need a similar procedure, called *narrowings*, to ensure termination.
- The only difference:

$$\forall x, y \in L : (y \sqsubseteq x) \implies (y \sqsubseteq (x \vartriangle y) \sqsubseteq x)$$

Not the dual of widening

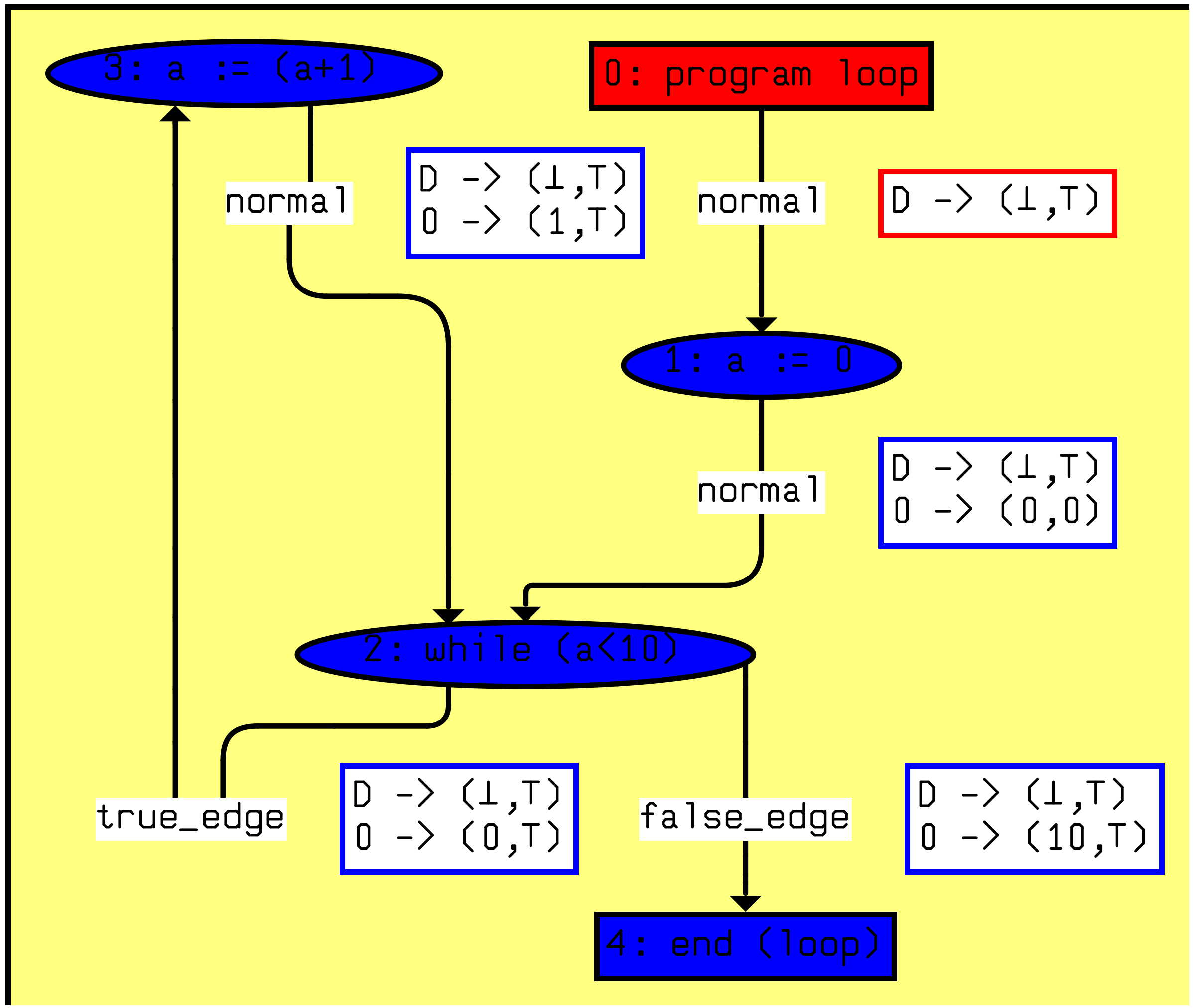
- The difference:

$$\forall x, y \in L : (y \sqsubseteq x) \implies (y \sqsubseteq (x \mathbin{\triangle} y) \sqsubseteq x)$$

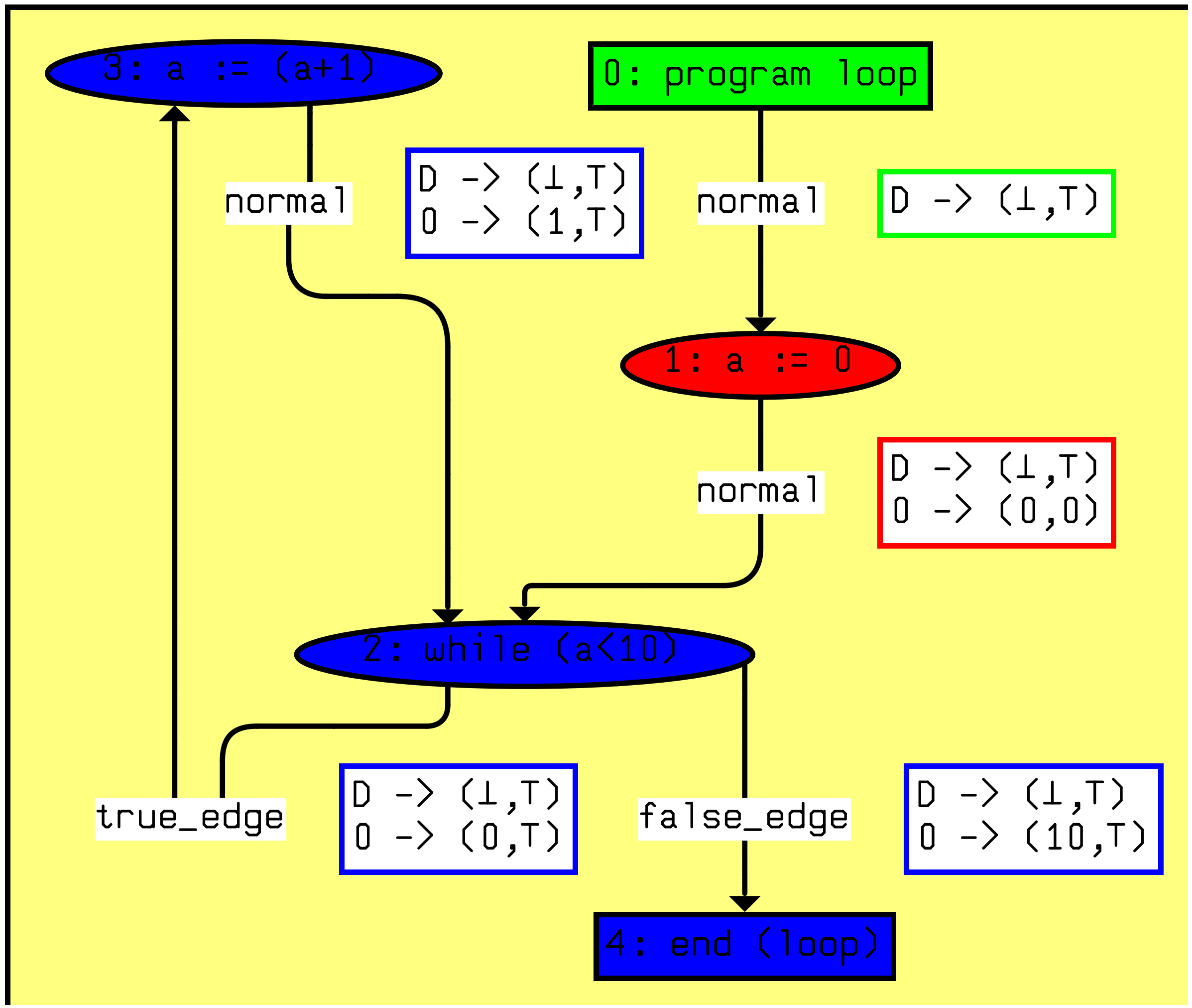
means it is not a lower bound operator.

- While narrowings keep the sequence in Red, the dual of widenings would step out of Red and eventually into Ext.

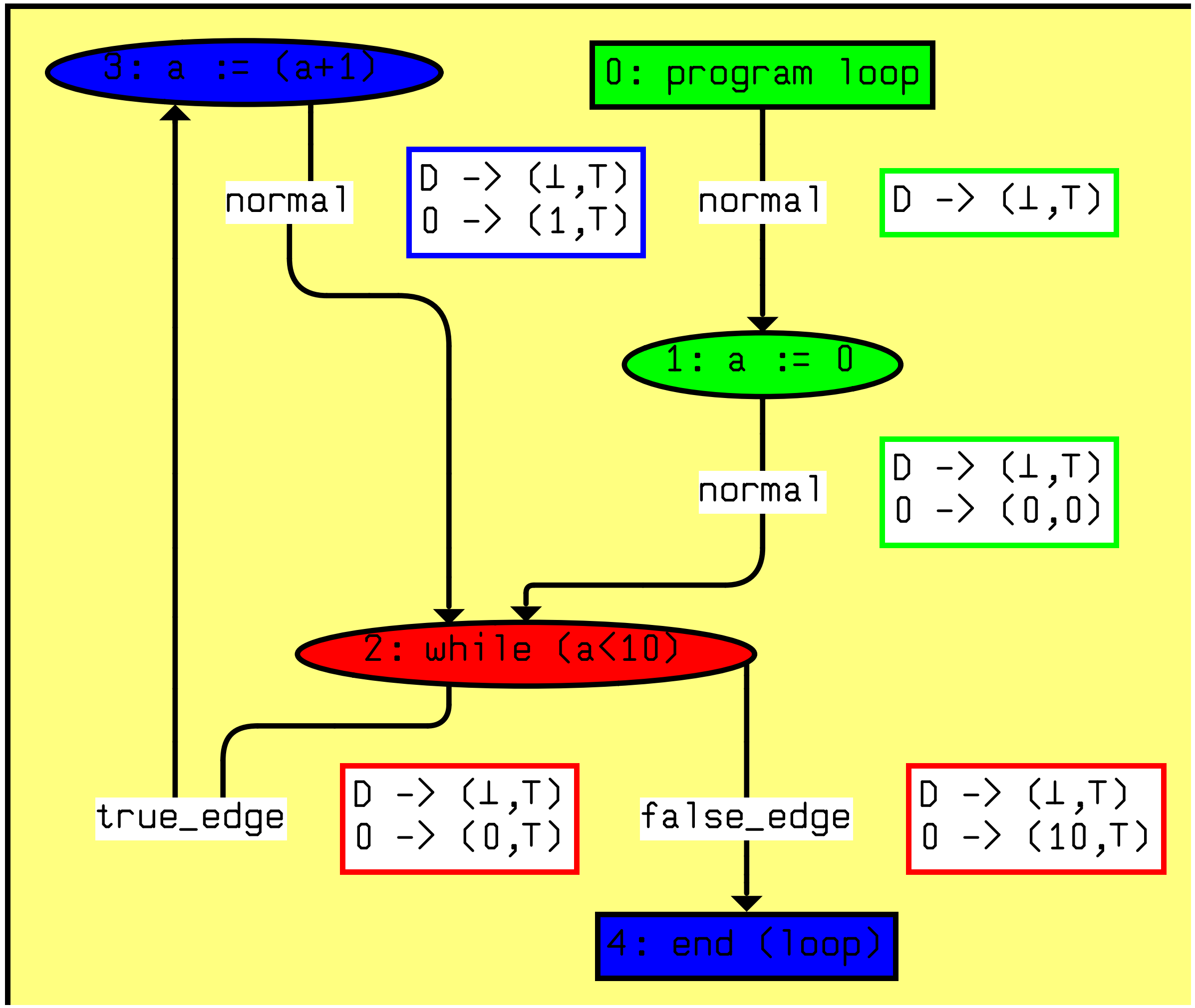
Fine-tuning the result



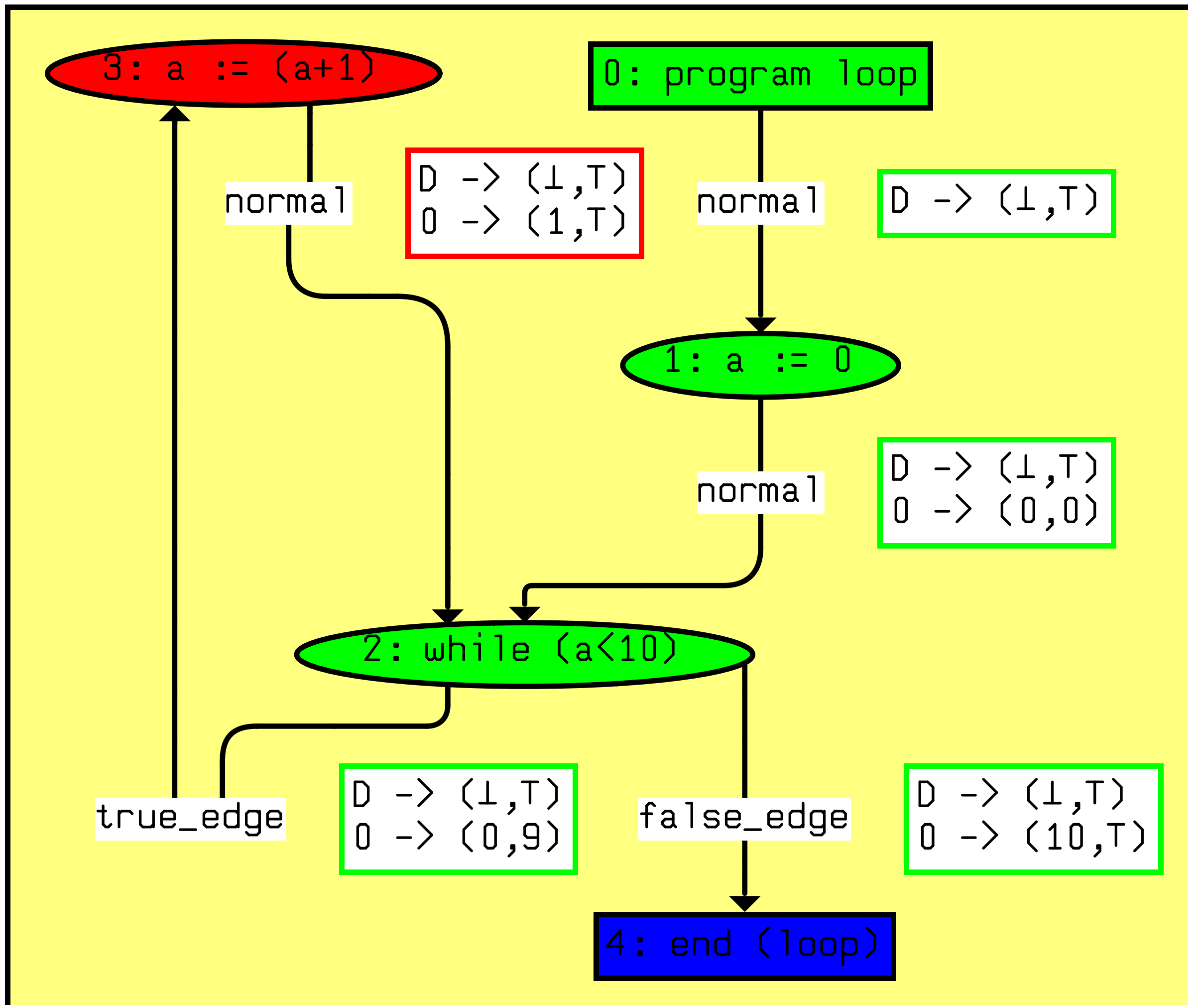
Fine-tuning the result



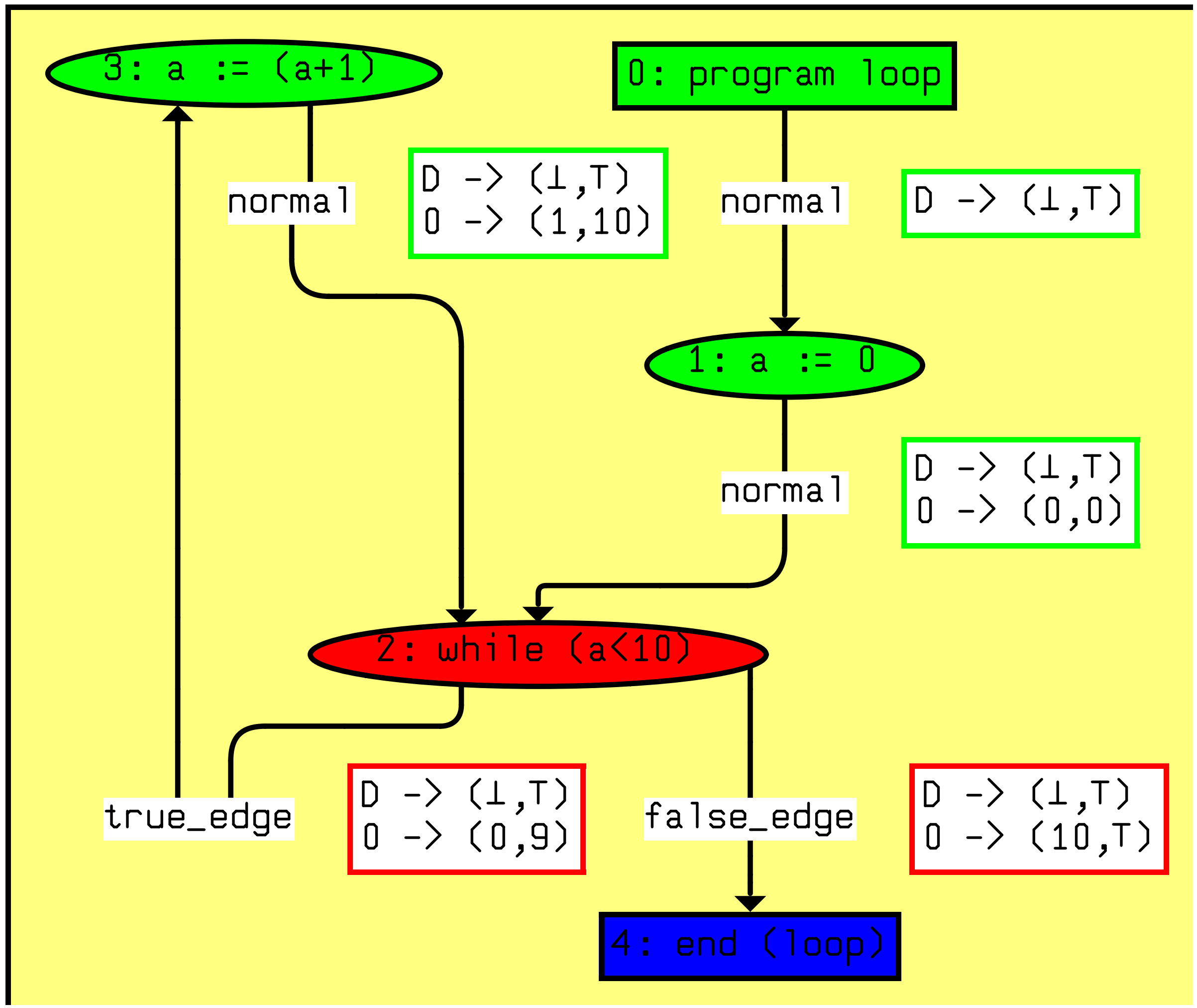
Fine-tuning the result



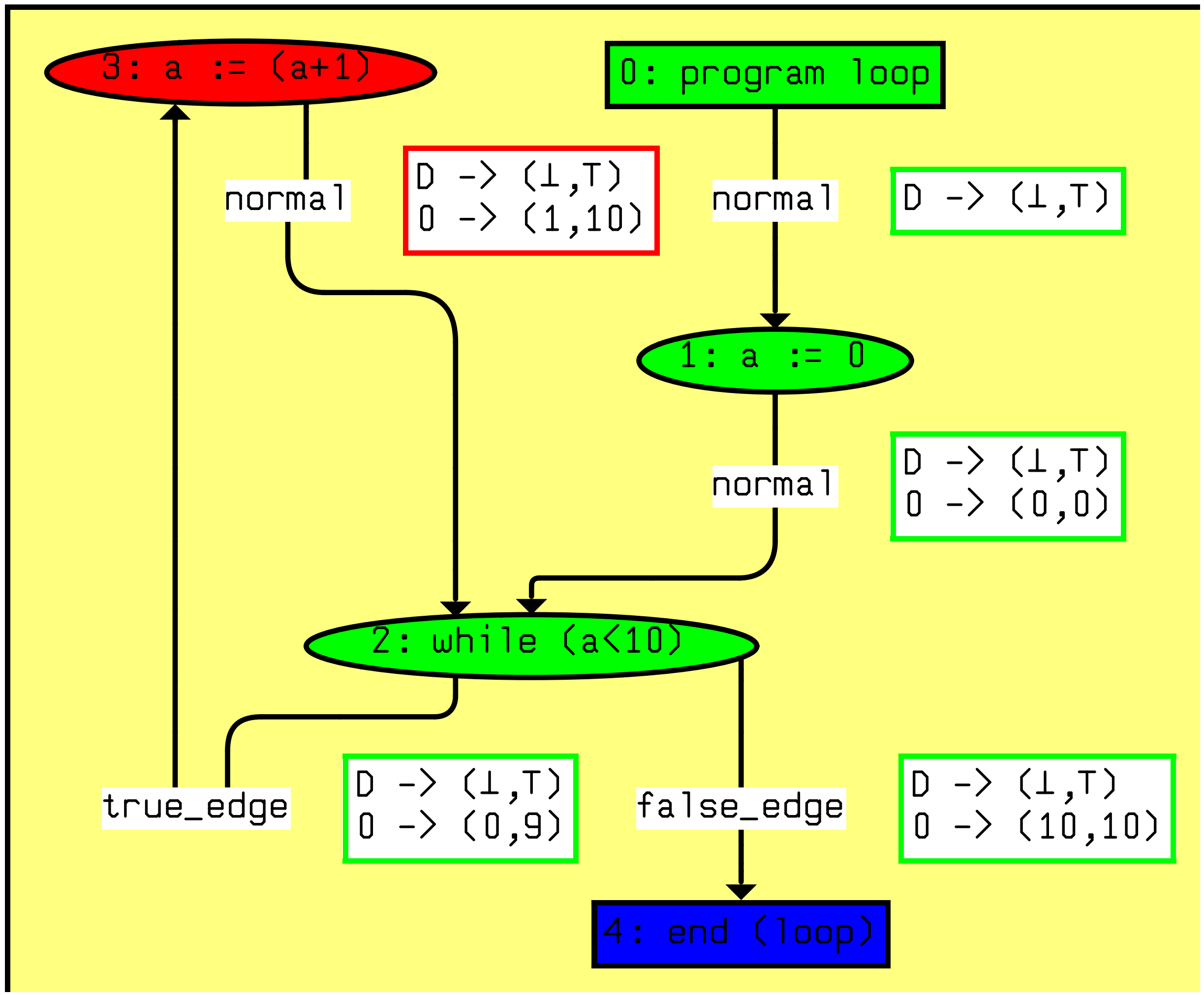
Fine-tuning the result



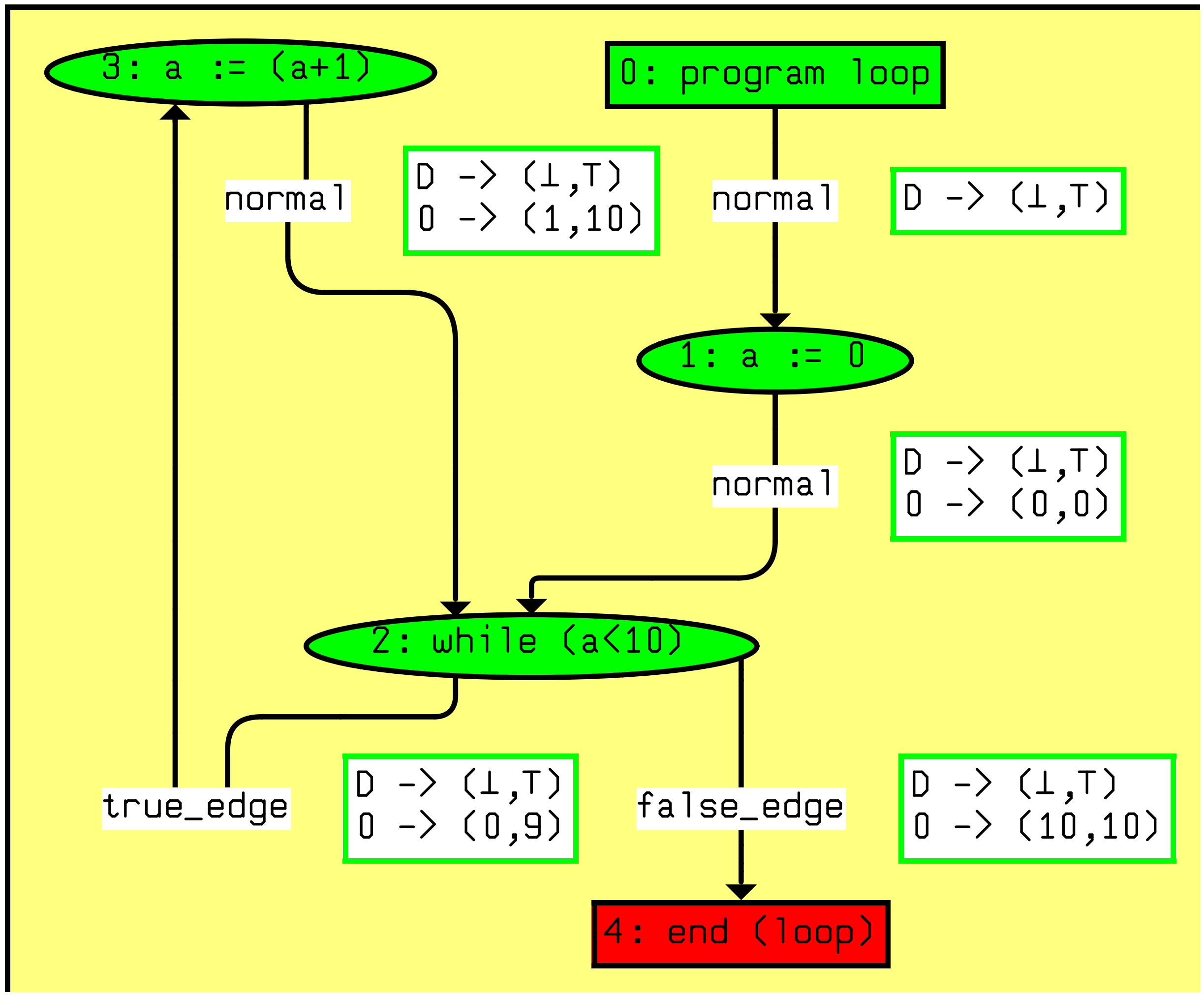
Fine-tuning the result



Fine-tuning the result



Fine-tuning the result



Examples

Questions