Computationally Secure Information Flow

Peeter Laud
Universität des Saarlandes

supported by Deutsche Forschungsgemeinschaft, Graduate Studies Program #623, "Quality Guarantees for Computer Systems"

Introduction

Given program P, let the set of its variables be $\mathbf{Var}$.

$\mathbf{Var} = \mathbf{Var}_{\mathrm{HighSec}} \uplus \mathbf{Var}_{\mathrm{LowSec}}$ — high and low security/secrecy variables.

A program P is secure if

- <u>one</u> cannot find anything about the initial values of $\mathbf{Var}_{\mathrm{HighSec}}$ from the final values of $\mathbf{Var}_{\mathrm{LowSec}}$;

"X cannot find anything", X might be

- a function
- probabilistic polynomial-time algorithm

I'm going to present an analysis for a language which contains a cryptographic primitive — the encryption.

Structure of the talk

- Syntax
- Semantics
- Confidentiality definition
- Analysis

Syntax

The WHILE-language. A statement s is one of

- Assignment $x = o(x_1, \ldots, x_k)$ $x, x_1, \ldots, x_k \in \mathbf{Var}$, $o \in \mathbf{Op}$
- Sequence $s_1; s_2$
- Branch $\texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2$ $b \in \mathbf{Var}$
- Loop $\texttt{while}\ b\ \texttt{do}\ s_1$ $b \in \mathbf{Var}$

Represent programs as flowcharts

- A node is an assignment or a test
- Introduce extra *start*- and *end*-nodes.

Semantics

- Denotational semantics — maps program's initial state to its final state.
- Usual type of denotational semantics: $\mathbf{State} \to \mathbf{State}_\perp$.
  $\mathbf{Val}$ — the set of values; $\mathbf{State} = \mathbf{Var} \to \mathbf{Val}$; $\mathbf{State}_\perp = \mathbf{State} \uplus \{\perp\}$.
- The semantics of $o \in \mathbf{Op}$ with arity k: $[\![o]\!] : \mathbf{Val}^k \to \mathbf{Val}$
  Program semantics defined with help of a fixed-point operation
- We want to have distinguished $\mathcal{E}nc, \mathcal{G}en \in \mathbf{Op}$.
  $\mathcal{E}nc$ is binary, $\mathcal{G}en$ is nullary.
  What kind of functions should $[\![\mathcal{E}nc]\!]$ and $[\![\mathcal{G}en]\!]$ be?

Requirements for the encryption scheme

Encryption scheme (Gen, Enc) is *which-key and repetition concealing* $\Longleftrightarrow$ $\forall$ PPT algorithm $\mathcal{A}^{(\cdot),(\cdot)}$

$$\Pr[k, k' \leftarrow Gen(\mathbf{1}^n) : \mathcal{A}^{Enc(\mathbf{1}^n,k,\cdot),Enc(\mathbf{1}^n,k',\cdot)}(\mathbf{1}^n) = 1] -$$
$$\Pr[k \leftarrow Gen(\mathbf{1}^n) : \mathcal{A}^{Enc(\mathbf{1}^n,k,\mathbf{0}^{\ell(n)}),Enc(\mathbf{1}^n,k,\mathbf{0}^{\ell(n)})}(\mathbf{1}^n) = 1]$$

is negligible in n.

$[\![\mathcal{E}nc]\!]$ and $[\![\mathcal{G}en]\!]$ must be

- families of functions (indexed by the *security parameter* $n \in \mathbb{N}$);
- probabilistic.
- Also, **Val** must be the set of bit-strings.

The type of semantics

Let $\ell \in \mathbb{Z}[n]$;

$\mathbf{Val}_n = \{0,1\}^{\ell(n)}$;

$\mathbf{State}_n = \mathbf{Var} \to \mathbf{Val}_n$

Type of semantics:

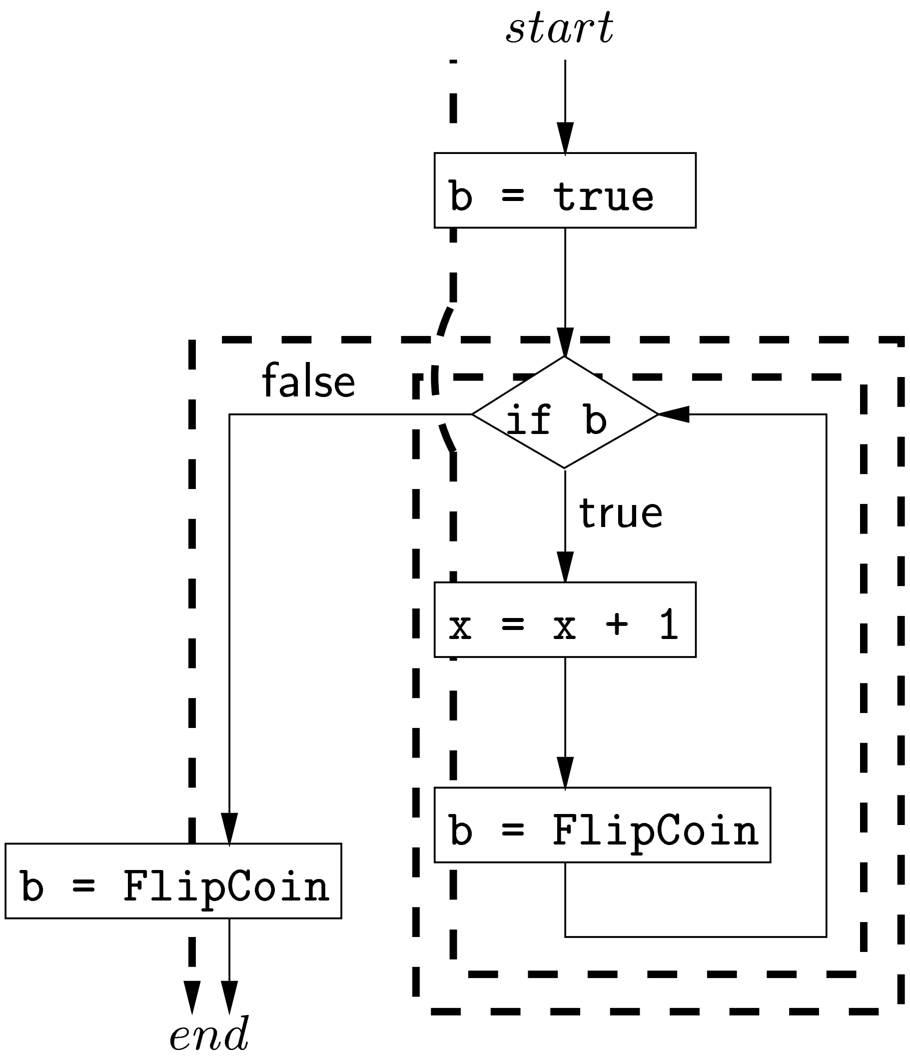
$\prod_{n \in \mathbb{N}} (\mathbf{State}_n \to \mathcal{D}(\mathbf{State}_{n\perp}))$

- Should take into account only *polynomially long computations*.

$\mathbf{Path} \subseteq Node^*$ — the set of paths in the flowchart from *start* to *end*.

Type of semantics:

$[\![\mathsf{P}]\!]_{\mathrm{Path}} : \mathbf{Path} \to \prod_{n \in \mathbb{N}} (\mathbf{State}_n \to \mathcal{D}(\mathbf{State}_{n\perp}))$.



Denote the marked path by P.
$[\![\mathsf{P}]\!]_{\mathrm{Path}}(P)$ maps each

$$\left\{\begin{array}{l} x \mapsto x_{\mathrm{init}} \\ b \mapsto b_{\mathrm{init}} \end{array}\right\}$$

to

$$\frac{1}{8}\left\{\begin{array}{l} x \mapsto x_{\mathrm{init}} + 2 \\ b \mapsto \mathsf{false} \end{array}\right\} + \frac{1}{8}\left\{\begin{array}{l} x \mapsto x_{\mathrm{init}} + 2 \\ b \mapsto \mathsf{true} \end{array}\right\} + \frac{3}{4}\perp.$$

Polynomially long computations

$\mathcal{S} : \mathbb{N} \to \mathcal{P}(\mathbf{Path})$ is *polynomially long*, $\Longleftrightarrow$

$$\exists q \in \mathbb{Z}[n] \text{ such that } \forall n \in \mathbb{N},\ \forall P \in \mathcal{S}(n) : |P| \leq q(n).$$

$\mathcal{PLP}$: the set of polynomially long path-sets.
Ordered pointwise, by inclusion.

$$[\![\mathsf{P}]\!] : \mathcal{PLP} \to \prod_{n \in \mathbb{N}} (\mathbf{State}_n \to \mathcal{D}(\mathbf{State}_{n\perp}))$$

$$[\![\mathsf{P}]\!](\mathcal{S})(S_{n,s}) = \left\{ \begin{array}{l} S_n \mapsto \sum_{P \in \mathcal{S}(n)} [\![\mathsf{P}]\!]_{\mathrm{Path}}(P)(S_{n,s})(S_n) \\ \perp \ \mapsto \text{what remains over } \ldots \end{array} \right\}$$

here $n \in \mathbb{N}$, $S_{n,s}, S_n \in \mathbf{State}_n$.

Recoverability

$v \in \mathbf{Var}$; $Y \subseteq \mathbf{Var}$; $\{D_{n,s}\}_{n\in\mathbb{N}} \in \{\mathcal{D}(\mathbf{State}_n)\}_{n\in\mathbb{N}}$.

Let the initial state of P be distributed according to $\{D_{n,s}\}_{n\in\mathbb{N}}$.
Initial value of v is *recoverable* from final values of Y, $\Longleftrightarrow$

There is something about v,	$\exists$ pol.-time computable predicate $\{B_n\}_{n\in\mathbb{N}}$ over $\{\mathbf{Val}_n\}_{n\in\mathbb{N}}$
such that if we let the program run long enough	$\exists\ \mathcal{S}_0 \in \mathcal{PLP}$
or even longer (insensitivity to timing)	$\forall\ \mathcal{S} \geq \mathcal{S}_0$ (where $\mathcal{S} \in \mathcal{PLP}$)

the final values of the variables in Y help to find the that something (the value of the predicate B) about the initial value of v.

$\forall$ PPT algorithm $\mathcal{B}$: $\exists$ PPT algorithm $\mathcal{A}$

$$\begin{aligned}\Pr[S_{n,s} \leftarrow D_{n,s}, S_n \leftarrow [\![\mathsf{P}]\!](\mathcal{S})(S_{n,s}) \ :\ \mathcal{A}(\mathbf{1}^n, S_n|_Y) = B_n(S_{n,s}(v))] - \\ \Pr[S_{n,s} \leftarrow D_{n,s} \ :\ \mathcal{B}(\mathbf{1}^n) = B_n(S_{n,s}(v))]\end{aligned}$$

is not negligible in n.

v *remains secret* if for each $Y \subseteq \mathbf{Var}$:

init. val. of v recoverable from final val. of $Y \Longrightarrow Y \cap \mathbf{Var}_{\mathrm{HighSec}} \neq \emptyset$.

Analysis-driven requirements

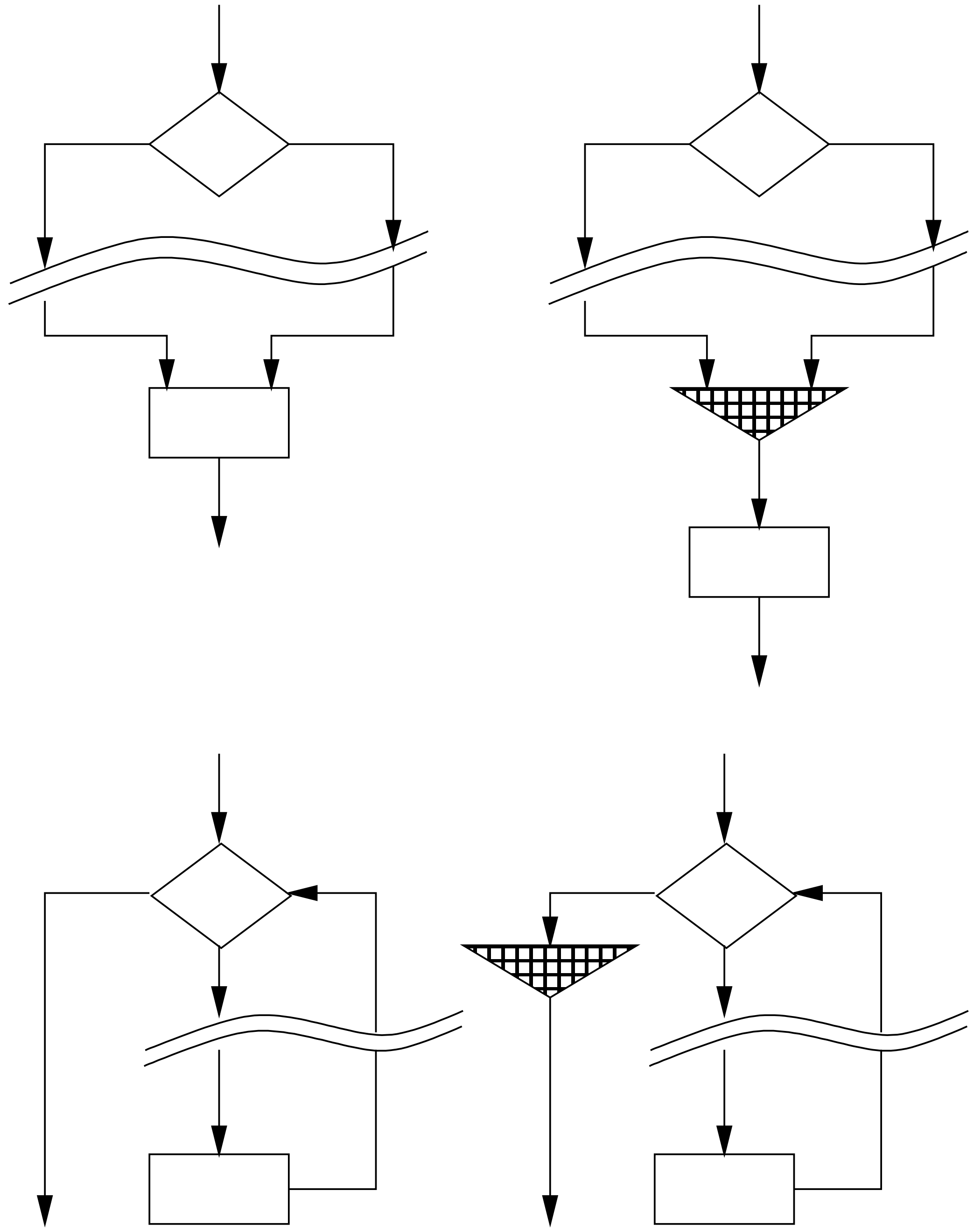
- $\forall o \in \mathbf{Op}\ \exists$ PPT algorithm $\mathcal{O}$ $\mathcal{O}(\mathbf{1}^n, \cdot) = [\![o]\!]_n(\cdot)$;
- may not mix keys and non-keys (imagine a simple type system);
- $\{D_{n,s}\}_{n\in\mathbb{N}}$ must be polynomial-time constructible;
- must know statically, which keys are equal and which are not.
- P runs in expected polynomial time.

```
k1 := gen_key()
if b then
  k2 := k1
else
  k2 := gen_key()
fi
x := enc(k1, y)
```

find `b` from `x`, `k2`

Control dependency regions

Add auxilliary *merge*-nodes on the borders of control dependency regions.



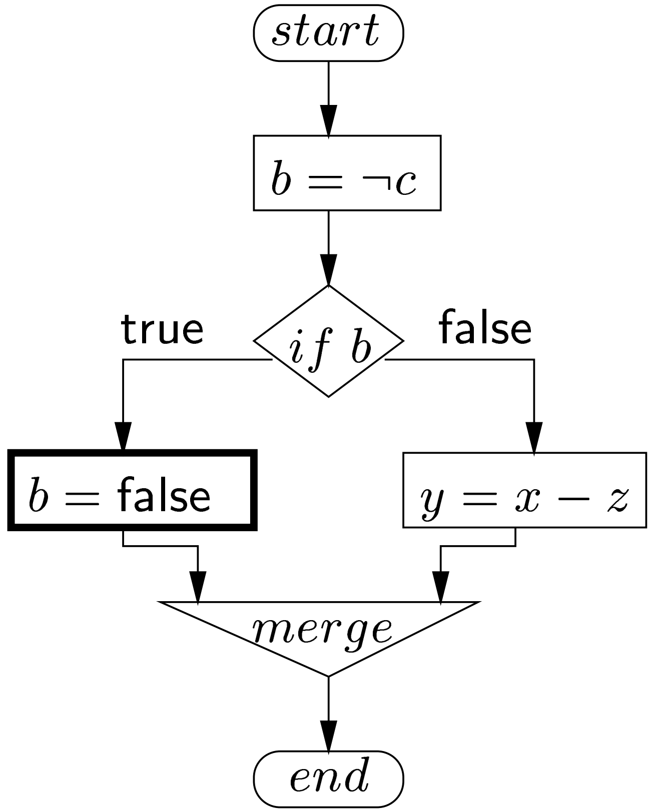
Analysis — structures

Fix $\{D_{n,s}\}_{n\in\mathbb{N}}$ and $v \in \mathbf{Var}$.
The analysis $\mathcal{W}$ has type
$Node \to \mathcal{P}_{\mathrm{U}}(\mathcal{P}(\widetilde{\mathbf{Var}}))$, where
$\widetilde{\mathbf{Var}} = \mathbf{Var} \cup Node_{if}$.

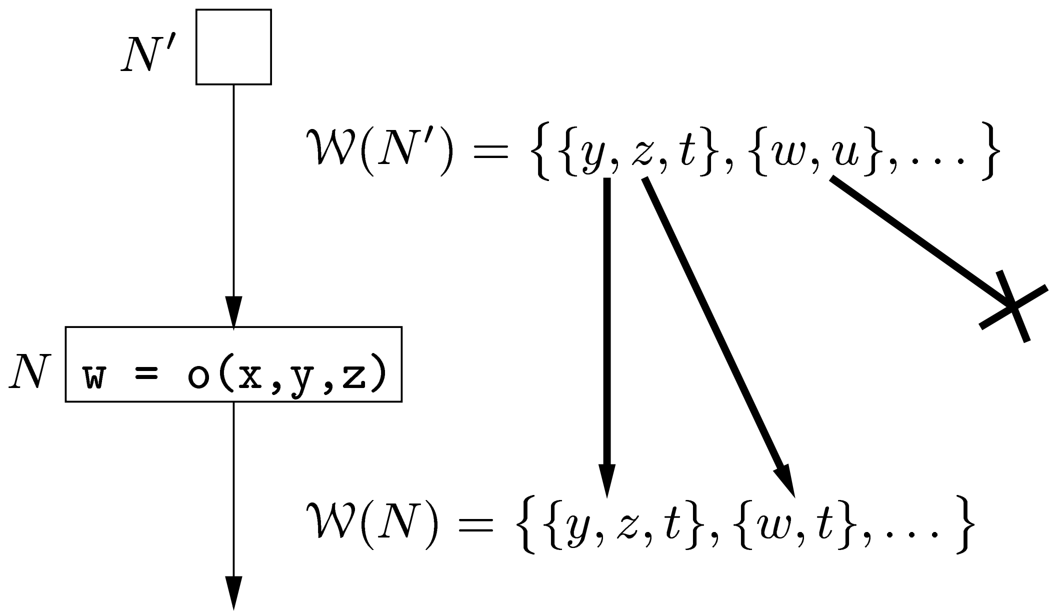
$Y \in \mathcal{W}(N)$ means: at node N, the initial value of v may be recoverable

- from the values of variables in $Y \cap \mathbf{Var}$;
- additionally taking into account, that the control flow has reached the node N because of choices made at nodes $Y \cap Node_{if}$.

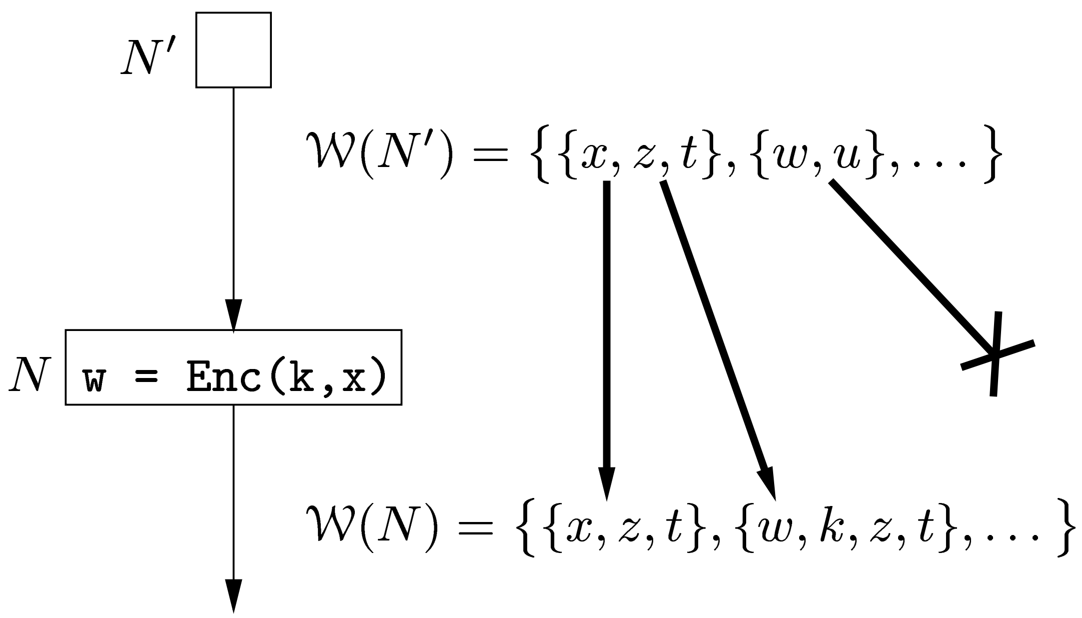
$\mathbf{Var} = \{b, c, x, y, z\}$



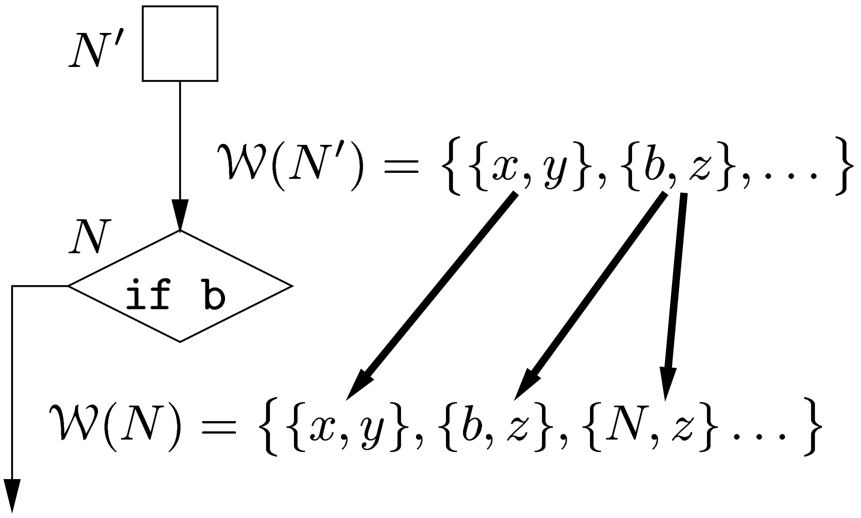
At the marked node:
c recoverable from $\{N_{if}\}$
y recoverable from $\{y\}$



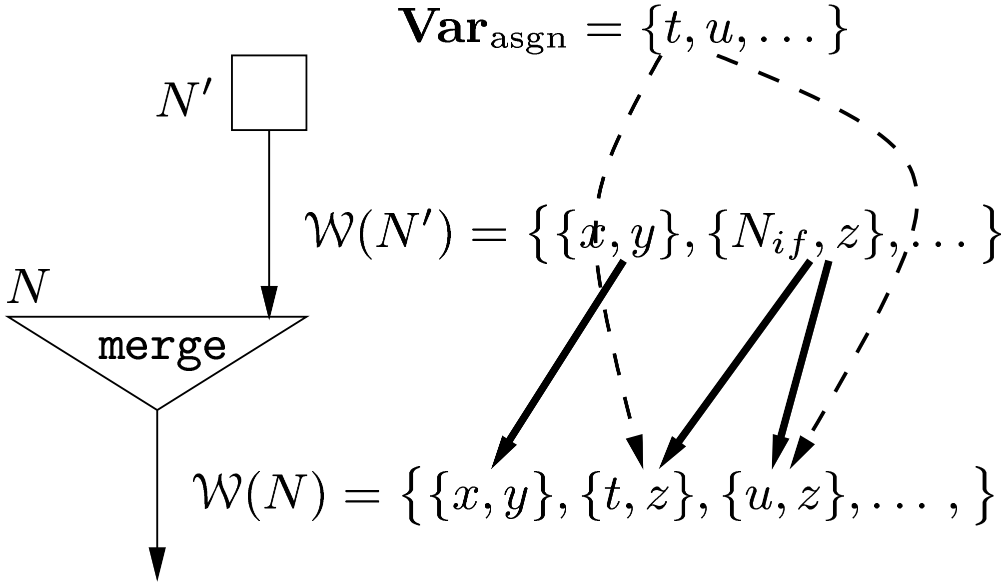
N'
$\mathcal{W}(N') = \{\{y, z, t\}, \{w, u\}, \dots\}$
N
w = o(x,y,z)
$\mathcal{W}(N) = \{\{y, z, t\}, \{w, t\}, \dots\}$



N'
$\mathcal{W}(N') = \{\{x, z, t\}, \{w, u\}, \dots\}$
N
w = Enc(k,x)
$\mathcal{W}(N) = \{\{x, z, t\}, \{w, k, z, t\}, \dots\}$



$\mathbf{Var}_{\text{asgn}}$ — the set of variables assigned between the corresponding N_{if} and *merge*.



For *start*-node: $\mathcal{W}$ is given by the structure of $\{D_{n,s}\}_{n\in\mathbb{N}}$.

Comparision with Denning's analysis

In def. of recoverability: v may be any polynomial time computable function of type $\prod_{n\in\mathbb{N}}(\mathbf{State}_n \to \{0,1\}^*)$.

Let $v = \mathbf{State}_n|_{\mathbf{Var}_{\mathrm{HighSec}}}$.

Let $\{D_{n,s}\}_{n\in\mathbb{N}}$ be such, that all variables are independent.
Without special handling of encryption, each $\mathcal{W}(N)$ has the form $\mathsf{cl}_{\mathrm{U}}\{\{x_1\},\dots,\{x_k\}\}$.

Identifying

$$\mathsf{cl}_{\mathrm{U}}\{\{x_1\},\dots,\{x_k\}\} \Longrightarrow \{x_1,\dots,x_k\} \Longrightarrow \mathcal{X}_{\{x_1,\dots,x_k\}} : \widetilde{\mathbf{Var}} \to \left\langle \begin{matrix}\{\ 0\ ,\ 1\ \}\\ \{\mathrm{low},\mathrm{high}\}\end{matrix}\right.$$

gives us Denning's information flow analysis in her terms (more or less; for the two-element lattice of security classes).