A 3-Valued Logic for the Specification and the Verification of Security Properties

Béchir Ktari

Université Laval
Québec, Canada

Foundations of Computer Security - FCS'03
LICS'03 Satellite Workshop

Outline

- Motivation
- Standard semantics
- 3-Valued semantics
- Extensions
- Future Work

μ-calculus

- Syntax

$$\begin{array}{lcl} \psi & ::= & \mathsf{tt} \mid \mathsf{ff} \mid \neg\psi \mid \psi \vee \psi' \mid \psi \wedge \psi' \mid \texttt{<}K\texttt{>}\psi \mid \texttt{[}K\texttt{]}\,\psi \mid X \mid \mu X.\psi \mid \nu X.\psi \\ K & ::= & \{a_i \mid i = 1..n\} \mid K - K' \\ a_i & ::= & \textit{Critical Functions} \end{array}$$

- A state satisfies the formula $\texttt{<}K\texttt{>}\psi$ if it can evolve to some state obeying ψ by performing an action from K.
- A state satisfies the formula $\texttt{[}K\texttt{]}\,\psi$ if, after the performance of any action in K, the result state satisfies ψ.
- Fixpoint operators allow the specification of recursive behaviors and to reason about all the states of the model.

μ-calculus

- Abbreviations:
  - $\text{always}(\psi) = \nu X.\psi \wedge [\mathsf{all}]\, X$
  - $\text{eventually}(\psi) = \mu X.\psi \vee \langle\mathsf{all}\rangle X$
  - $\text{never}(\psi) = \neg\text{eventually}(\psi)$
  - $\text{loop}(a) = \nu X.\langle a\rangle X$
- Examples of properties:
  - $[\mathit{readPassword}]\langle \mathit{checkPassword}\rangle\mathsf{true}$
  - never($\langle \mathit{formatDisk}, \mathit{send}\rangle\mathsf{tt}$)
  - always($[\mathit{readFile}]$ never(do(send)))
  - always($[\mathit{openSocket}]$ eventually($\langle \mathit{closeSocket}\rangle\mathsf{true}$))
  - never(loop($\mathit{createProcess}$))

μ-calculus

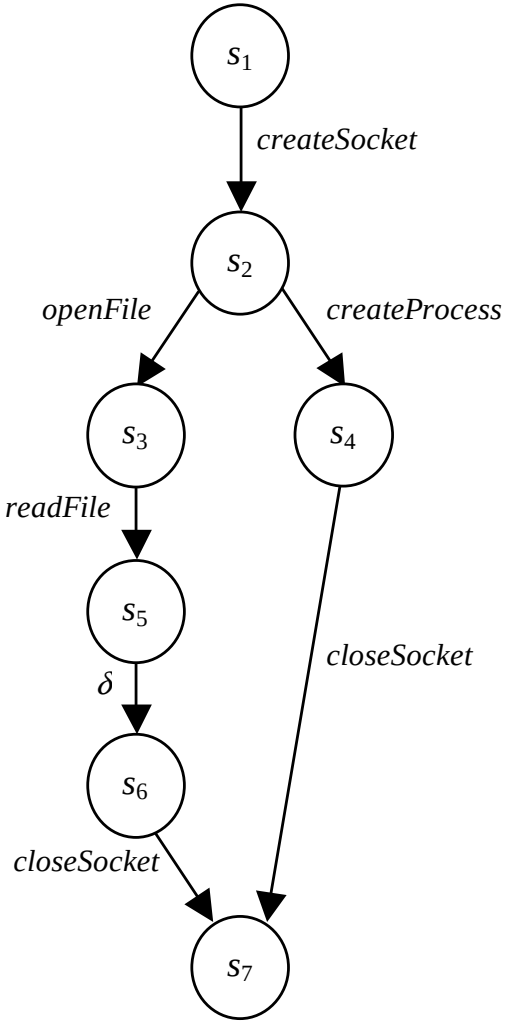
- Model: Labelled Transition System.
- Semantics: $[\![.]\!]_e^M : \mathcal{F} \to 2^{\mathcal{S}}$

$$
\begin{array}{ll}
[\![\mathsf{tt}]\!]_e^M & = \mathcal{S} \\
[\![\mathsf{ff}]\!]_e^M & = \emptyset \\
[\![X]\!]_e^M & = e(X) \\
[\![\neg\psi]\!]_e^M & = \overline{[\![\psi]\!]_e^M} \\
[\![\psi_1 \vee \psi_2]\!]_e^M & = [\![\psi_1]\!]_e^M \cup [\![\psi_2]\!]_e^M \\
[\![\psi_1 \wedge \psi_2]\!]_e^M & = [\![\psi_1]\!]_e^M \cap [\![\psi_2]\!]_e^M \\
[\![<K>\psi]\!]_e^M & = \{s \mid (\exists s' \mid s' \in [\![\psi]\!]_e^M : (\exists a \mid a \in K : s \xrightarrow{a} s'))\} \\
[\![\,[K]\,\psi]\!]_e^M & = \{s \mid (\forall s' \mid : (\forall a \mid a \in K : s \xrightarrow{a} s' \Rightarrow s' \in [\![\psi]\!]_e^M))\} \\
[\![\mu X.\psi]\!]_e^M & = \bigcap\{Q \subseteq \mathcal{S} \mid [\![\psi]\!]_{e[X \mapsto Q]}^M \subseteq Q\} \\
[\![\nu X.\psi]\!]_e^M & = \bigcup\{Q \subseteq \mathcal{S} \mid Q \subseteq [\![\psi]\!]_{e[X \mapsto Q]}^M\}
\end{array}
$$

- Model-Checker: True/False

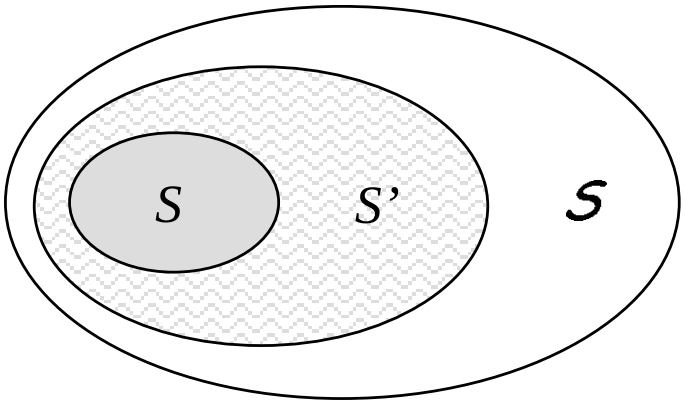
Model with δ-transitions

- Name of some function calls unavailable statically.
- Pointers, functions pointers, etc.
- 3-valued semantics: allows to reason under uncertainty (lack of information).



Toward a 3-Valued Semantics

- New meaning function: $[\![.]\!]_e^M : \mathcal{F} \rightarrow 2^{\mathcal{S}} \times 2^{\mathcal{S}}$
- Semantically, formulae of the logic correspond to sets of states for which they are true (S) and to sets of states for which they may be true (S'), including those that are true.



  The remaining states in $\mathcal{S}$ do not satisfy the formulae.

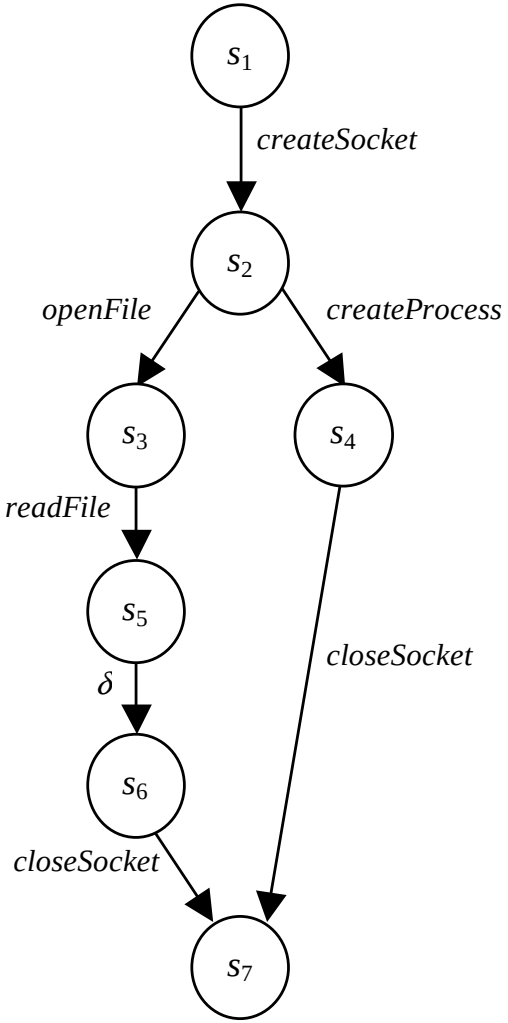
- Model-Checker: True/False/May be

3-Valued Semantics - First Version

- $[\![\mathsf{tt}]\!]^M_e = (\mathcal{S}, \mathcal{S})$
  $[\![\mathsf{ff}]\!]^M_e = (\emptyset, \emptyset)$
  $[\![X]\!]^M_e = e(X)$
  $[\![\neg\psi]\!]^M_e = (\overline{\pi_2([\![\psi]\!]^M_e)}, \overline{\pi_1([\![\psi]\!]^M_e)})$
  $[\![\psi_1 \vee \psi_2]\!]^M_e = (\pi_1([\![\psi_1]\!]^M_e) \cup \pi_1([\![\psi_2]\!]^M_e), \pi_2([\![\psi_1]\!]^M_e) \cup \pi_2([\![\psi_2]\!]^M_e))$
  $[\![\psi_1 \wedge \psi_2]\!]^M_e = (\pi_1([\![\psi_1]\!]^M_e) \cap \pi_1([\![\psi_2]\!]^M_e), \pi_2([\![\psi_1]\!]^M_e) \cap \pi_2([\![\psi_2]\!]^M_e))$
  $[\![\langle K \rangle\ \psi]\!]^M_e = (\ \{s \mid (\exists s' \mid s' \in \pi_1([\![\psi]\!]^M_e) : (\exists a \mid a \in K : s \xrightarrow{a} s'))\},$
  $\{s \mid (\exists s' \mid s' \in \pi_2([\![\psi]\!]^M_e) : (\exists a \mid a \in K_\delta : s \xrightarrow{a} s'))\}$
  $)$
  $[\![[K]\ \psi]\!]^M_e = (\ \{s \mid (\forall s' \mid : (\forall a \mid a \in K_\delta : s \xrightarrow{a} s' \Rightarrow s' \in \pi_1([\![\psi]\!]^M_e)))\},$
  $\{s \mid (\forall s' \mid : (\forall a \mid a \in K : s \xrightarrow{a} s' \Rightarrow s' \in \pi_2([\![\psi]\!]^M_e)))\}$
  $)$
  $[\![\mu X.\psi]\!]^M_e = \bigcap\{(Q, Q') \subseteq (\mathcal{S} \times \mathcal{S}) \mid [\![\psi]\!]^M_{e[X \mapsto (Q,Q')]} \subseteq (Q, Q')\}$
  $[\![\nu X.\psi]\!]^M_e = \bigcup\{(Q, Q') \subseteq (\mathcal{S} \times \mathcal{S}) \mid (Q, Q') \subseteq [\![\psi]\!]^M_{e[X \mapsto (Q,Q')]}\}$

Examples

- Ex1: [*createProcess*]<*closeSocket*>tt
- Ex2: <all><*closeSocket*>tt



3-Valued Semantics - Final Version

- $$\vdots$$

$$[\![\langle K \rangle\ \psi]\!]_e^M = (\ \{s \mid (\exists s' \mid s' \in \pi_1([\![\psi]\!]_e^M) : (\exists a \mid a \in K : s \xrightarrow{a} s'))\} \cup$$
$$\{s \mid (K = \mathsf{all}) \wedge (\exists s' \mid s' \in \pi_1([\![\psi]\!]_e^M) : s \xrightarrow{\delta} s')\},$$
$$\{s \mid (\exists s' \mid s' \in \pi_2([\![\psi]\!]_e^M) : (\exists a \mid a \in K_\delta : s \xrightarrow{a} s'))\}$$
$$)$$

$$[\![[K]\ \psi]\!]_e^M = (\ \{s \mid (\forall s' \mid : (\forall a \mid a \in K_\delta : s \xrightarrow{a} s' \Rightarrow s' \in \pi_1([\![\psi]\!]_e^M)))\},$$
$$\{s \mid (\forall s' \mid : (\forall a \mid a \in K : s \xrightarrow{a} s' \Rightarrow s' \in \pi_2([\![\psi]\!]_e^M)))\} \cap$$
$$\{s \mid (K = \mathsf{all}) \ \Rightarrow\ (\forall s' \mid : s \xrightarrow{\delta} s' \Rightarrow s' \in \pi_2([\![\psi]\!]_e^M))\}$$
$$)$$

- $$\vdots$$

Properties

The semantics preserves the classical properties of μ-calculus:

$$\begin{aligned} \neg\neg\psi &\equiv \psi \\ \neg(\psi_1 \vee \psi_2) &\equiv \neg\psi_1 \wedge \neg\psi_2 \\ [K]\,\psi &\equiv \neg\langle K\rangle\neg\psi \\ \mu X.\psi &\equiv \neg\nu X.\neg\psi[\neg X/X] \end{aligned}$$

Tableau-based Proof System

- Local Model Checking: Does not require that every state in the model be examined.
- Two sequents:
  1. $H \vdash s \in \psi$
  2. $H \vdash s \approx \psi$

  where H is a mapping in $[\mathcal{V} \mapsto 2^{\mathcal{S}}]$, s is a state and ψ is a formula.
- Three results: **Finiteness**, **Soundness** and **Completeness**.

Toward a more Efficient Semantics

- Let's
  - $M = \langle \{s_0, s_1\}, \{a, b, c\}, s_0 \xrightarrow{\delta} s_1, s_0 \rangle$ be a model, and
  - $\psi = \texttt{<}a\texttt{>tt} \vee \texttt{<-}a\texttt{>tt}$ be a formula.
- By definition, we have the following:

$$[\![\texttt{<}a\texttt{>tt} \vee \texttt{<-}a\texttt{>tt}]\!]_e^M = (\pi_1([\![\texttt{<}a\texttt{>tt}]\!]_e^M) \cup \pi_1([\![\texttt{<-}a\texttt{>tt}]\!]_e^M), \\ \pi_2([\![\texttt{<}a\texttt{>tt}]\!]_e^M) \cup \pi_2([\![\texttt{<-}a\texttt{>tt}]\!]_e^M))$$

  It is easy to see that: $[\![\texttt{<}a\texttt{>tt}]\!]_e^M = [\![\texttt{<-}a\texttt{>tt}]\!]_e^M = (\emptyset, \{s_0\})$

  It follows that: $[\![\texttt{<}a\texttt{>tt} \vee \texttt{<-}a\texttt{>tt}]\!]_e^M = (\emptyset, \{s_0\})$
- However, the formula ψ expresses the two following facts:
  1. A state has an a-transition;
  2. A state has an "-a"-transition, i.e., an $\{b, c\}$-transition.

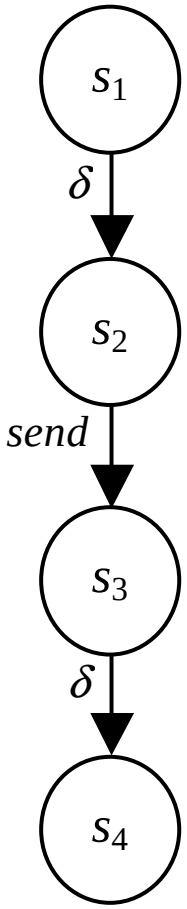
  meaning that the state s_0 should satisfy the formula.

Outline

- Motivation
- Standard semantics
- 3-Valued semantics
- Extensions
  1. Naming δ-transition.
  2. Typing δ-transition.
- Future Work

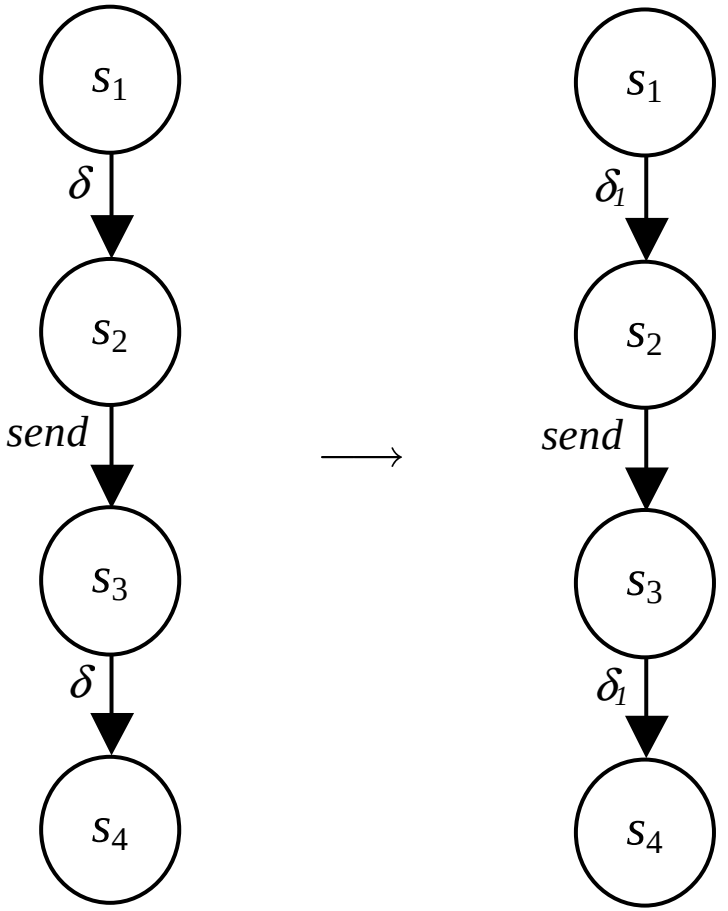
Extension 1

```
extern int openFile(char*);
extern void closeFile(int);
extern int send(int, char*);

int main()
{
        int socket;
        int file;
        char fileName[1000];
        int (*f)(char *);
        ...
        f("f1.ex");
        send( socket, "done" );
        f("f2.ex");
        return 0;
}
```



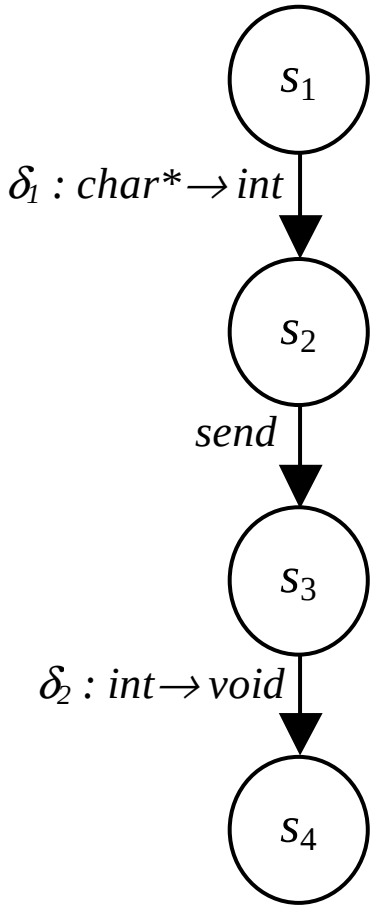
Extension 1

```
extern int openFile(char*);
extern void closeFile(int);
extern int send(int, char*);

int main()
{
        int socket;
        int file;
        char fileName[1000];
        int (*f)(char *);
        ...
        f("f1.ex");
        send( socket, "done" );
        f("f2.ex");
        return 0;
}
```



Extension 2

```
...
int main()
{
        int socket;
        int file;
        char fileName[1000];
        int (*f)(char*);
        void (*g)(int);
        ...
        file = f("f1.ex");
        send( socket, "done" );
        g(file);
        return 0;
}
```



Future Work

- More expressivity: Logic to specify properties involving data.
  - Examples:
    - never(<`read`(x, ρ_1^s)>eventually(<`write`(x, ρ_2^p)>`tt`))
    - always([`write`$(f, dir^\alpha($`tempdir`$))$]eventually(<`delete`$(\_, file^\alpha(f))$>`tt`))
  - 3-Valued semantics should help.
  - *Modal transition systems: A foundation for three-valued program analysis*, [Jagadeesan and Schmidt, 2001].
- Model?
  Use of the certifying-compilation technology in order to generate a model suitable for security analysis.
  - Model-Carrying Code.
  - Effects Type Systems.
  - *Models for Security Policies in Proof-Carrying Code*, [Appel and Felten, 2001].