Calculational design of a static dependency analysis

Patrick Cousot
New York University, Courant Institute of Mathematics, Computer Science
pcousot@cs.nyu.edu  cs.nyu.edu/~pcousot
Motivation
Dependency

Dependency is prevalent in computer science:

- Non-interference (confidentiality, integrity)
- Security, privacy
- Slicing
- Temporal dependencies in synchronous languages (Lustre, Signal, etc.)
- etc.

The existing definitions

- are postulated a priori (par exemple Cheney, Ahmed, and Acar, 2011; D. E. Denning and P. J. Denning, 1977),
- without semantics justifications (except Assaf, Naumann, Signoles, Totel, and Tronel, 2017 (“hyper-collecting semantics”), Urban and Müller, 2018 on program exit uniquely)

We are interested in principles, in soundness proofs, not so much in a new more powerful dependency analysis.
Structural fixpoint trace semantics
Program syntax

- C statements limited to integers, assignments, statement lits, conditionals, iterations
- Programs are labelled to designate program points
  - at\([S]\): entry program point of \(S\) starts;
  - after\([S]\): normal exit program point of \(S\);
  - in\([S]\): reachable program points of \(S\) (excluding after\([S]\));
  - break-to\([S]\): breaking point when \(S\) contains a break; to exit a loop (then escape\([S]\) = tt);
Execution traces

- Program:

\[ \ell_1 \ x = 0 \ ; \text{while} \ \ell_2 (tt) \ \{ \ \ell_3 \ x = x+1 ; \ \} \ \ell_4 \]

- Infinite execution trace:

\[ \ell_1 \ x = 0 = 0 \rightarrow \ell_2 \ \text{tt} \rightarrow \ell_3 \ x = x+1 = 1 \rightarrow \ell_2 \ \text{tt} \rightarrow \ell_3 \ x = x+1 = n \rightarrow \ell_2 \ \text{tt} \rightarrow \ell_3 \ x = x+1 = n+1 \rightarrow \ell_2 \ \ldots \]

- Trace: finite or infinite sequence of program points separated by action

\((x = A = \text{value}, B, \neg B, \text{et break} ;)\)
Value of a variable (and an expression)

- The value of a variable $x$ along a trace $\pi$ is the last assigned value (or 0 at initialization).

$$q(\pi^\ell) \xrightarrow{x = E = v} \ell' x \triangleq v$$
$$q(\pi^\ell \cdots \ell' x \triangleq q(\pi^\ell) \quad \text{otherwise}$$
$$q(\ell) x \triangleq 0$$

- Value of an arithmetic expression

$$\mathcal{A}[1] \rho \triangleq 1$$
$$\mathcal{A}[x] \rho \triangleq \rho(x)$$
$$\mathcal{A}[A_1 - A_2] \rho \triangleq \mathcal{A}[A_1] \rho - \mathcal{A}[A_2] \rho$$

- Same for boolean expressions.
Structural fixpoint prefix/maximal trace semantics $\widehat{\mathcal{S}}^*[S]$

- The prefix trace semantics $\widehat{\mathcal{S}}^*[S]$ is a relation between
  - an initialization trace $\pi_0$ at $[S]$ arriving at $[S]$, and
  - the prefix execution traces at $[S]\pi$ continuing this initialization by zero or more execution steps

- The maximal trace semantics $\widehat{\mathcal{S}}^{+\infty}[S]$ collects the maximal finite traces and the infinite traces obtained as limits of their prefixes.
Structural fixpoint definition of the prefix trace semantics (I)

- Assignment \( S ::= \ell \ x = A ; \) (where at\( S \) = \( \ell \))

\[
S^* [S] \triangleq \{ \langle \pi^\ell, \ell \rangle \mid \pi^\ell \in T^+ \} \cup \\
\{ \langle \pi^\ell, \ell \xrightarrow{x = A = \nu} \text{ after}[S] \rangle \mid \pi^\ell \in T^+ \land \nu = S[A]Q(\pi^\ell) \}
\]
Structural fixpoint definition of the prefix trace semantics (II)

- Iteration $S := \text{while } \ell \; (B) \; S_b$ (where $\text{at}[S] = \ell$):

\[
S^* [S] = \text{lfp}^c \mathcal{F}^*[S]
\]

\[
\mathcal{F}^*[\text{while } \ell \; (B) \; S_b](X) \triangleq \{ \langle \pi_1 \ell', \ell' \rangle \mid \pi_1 \ell' \in T^+ \land \ell' = \ell \}
\]

∪ \{ ⟨\pi_1 \ell', \ell'\pi_2 \ell' \xrightarrow{\neg(B)} \text{after}[S]⟩ \mid ⟨\pi_1 \ell', \ell'\pi_2 \ell'⟩ ∈ X \land
\]

\[
\mathcal{B}[B]\varrho(\pi_1 \ell'\pi_2 \ell') = \text{ff} \land \ell' = \ell \} \quad (a)
\]

∪ \{ ⟨\pi_1 \ell', \ell'\pi_2 \ell' \xrightarrow{B} \text{at}[S_b] \dashv \pi_3⟩ \mid ⟨\pi_1 \ell', \ell'\pi_2 \ell'⟩ ∈ X \land
\]

\[
\mathcal{B}[B]\varrho(\pi_1 \ell'\pi_2 \ell') = \text{tt} \land ⟨\pi_1 \ell'\pi_2 \ell' \xrightarrow{B}\text{at}[S_b], \pi_3⟩ ∈ S^*[S_b] \land \ell' = \ell \} \quad (b)
\]

\[
\mathcal{B}[B]Q(\pi_1 \ell'\pi_2 \ell') = \text{tt} \land ⟨\pi_1 \ell'\pi_2 \ell' \xrightarrow{B}\text{at}[S_b], \pi_3⟩ ∈ S^*[S_b] \land \ell' = \ell \} \quad (c)
\]

A definition of the form $d(\bar{x}) \triangleq \{ f(\bar{x}') \mid P(\bar{x}', \bar{x}) \}$ has the variables $\bar{x}'$ in $P(\bar{x}', \bar{x})$ bound to those of $f(\bar{x}')$ whereas $\bar{x}$ is free in $P(\bar{x}', \bar{x})$ since it appears neither in $f(\bar{x}')$ nor (by assumption) under quantifiers in $P(\bar{x}', \bar{x})$. The $\bar{x}$ of $P(\bar{x}', \bar{x})$ is therefore bound to the $\bar{x}$ of $d(\bar{x})$. 

Properties
Property

- A property is represented by a set of elements (those elements which have the property)
- Even integers: \(2\mathbb{Z} \triangleq \{2k \mid k \in \mathbb{Z}\}\)
- \(x\) has property \(P\) is \(x \in P\)
- Implication is \(P_1 \subseteq P_2\)
Semantic property

- The prefix trace semantics belongs to $\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty})$
- A semantics property belongs to $\wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}))$
- The abstraction

$$\langle \wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}), \subseteq \rangle \xrightarrow{\lambda Q \cdot \wp(Q)} \langle \wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}), \subseteq \rangle$$

provides trace properties (e.g. safety, liveness, etc.)
Dependency, informally
Dependency, informally

- At program point $\ell$, the variable $y$ depends upon the initial value $x_0$ of variable $x$ iff
  changing only $x_0$ will change the non-empty sequences of values $y_0, y_1, \ldots$ of $y$
  observed at $\ell$ whenever control reaches $\ell$

- Example: $\ell_0$ if (x=0) { y=x; $\ell_1$ } $\ell_2$
  - $y$ does not depend on $x$ neither at $\ell_0$ nor at $\ell_1$
  - $y$ depends on $x$ at $\ell_2$

- No need to distinguish between explicit and implicit dependencies
- Absence of observation is not an observation
- No timing channels
Dependency, formally
Observation of the sequence of values of a variable at a program point

- non-empty initialization trace $\pi_0 \in \mathbb{T}^+$
- non-empty continuation trace $\pi \in \mathbb{T}^{+\infty}$
- $\text{seqval}^\ell[y](\pi_0, \pi)$ is the sequence of values of the variable $y$ at program point $\ell$ along the trace $\pi$ continuing $\pi_0$

\[
\begin{align*}
\text{seqval}^\ell[y](\pi_0, \ell) & \triangleq q(\pi_0)y \\
\text{seqval}^\ell[y](\pi_0, \ell') & \triangleq \emptyset \\
\text{seqval}^\ell[y](\pi_0, \ell \xrightarrow{a} \ell'' \pi) & \triangleq q(\pi_0)y \cdot \text{seqval}^\ell[y](\pi_0 \xrightarrow{a} \ell'' \pi, \ell'' \pi) \\
\text{seqval}^\ell[y](\pi_0, \ell' \xrightarrow{a} \ell'' \pi) & \triangleq \text{seqval}^\ell[y](\pi_0 \xrightarrow{a} \ell'', \ell'' \pi)
\end{align*}
\]

- $\text{seqval}^\ell[y](\pi_0, \pi)$ is the empty sequence $\emptyset$ if $\ell$ never appears in $\pi$

(co-inductive definition for infinite traces).
Difference between sequences of values $\omega$ and $\omega'$

- Sequences that differ may have a common prefix but must eventually have a different value at some position in the sequences.

\[
\text{diff}(\omega, \omega') \triangleq \exists \omega_0, \omega_1, \omega'_1, \nu, \nu' . \omega = \omega_0 \cdot \nu \cdot \omega_1 \land \omega' = \omega_0 \cdot \nu' \cdot \omega'_1 \land \nu \neq \nu'
\]
Dependency, formally

- **Dependency property:**

\[ \mathcal{D}_{\text{diff}}^\ell(x, y) \triangleq \{ \Pi \in \wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}) \mid \exists \langle \pi_0, \pi_1 \rangle, \langle \pi'_0, \pi'_1 \rangle \in \Pi . \]

\((\forall z \in V \setminus \{x\} . \mathcal{G}(\pi_0)z = \mathcal{G}(\pi'_0)z) \land \)

\[ \text{diff}(\text{seqval}[\ell][y](\pi_0, \pi_1), \text{seqval}[\ell][y](\pi'_0, \pi'_1)) \}\]

- **y** depends on the initial value of **x** at program point \( \ell \) in program **P** is:

\[ \mathcal{S}^{+\infty}[P] \in \mathcal{D}_{\text{diff}}^\ell(x, y) \]

- **Lemma**

\[ \mathcal{S}^{+\infty}[P] \in \mathcal{D}_{\text{diff}}^\ell(x, y) \iff \mathcal{S}^*[P] \in \mathcal{D}_{\text{diff}}^\ell(x, y) \]
Value dependency abstraction
Abstraction en dépendance de données

- The abstraction of a semantic property $S \in \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{\infty})))$ into a value dependency property $\alpha^d(S) \in L \rightarrow \wp(\mathcal{V} \times \mathcal{V})$ is:

$$\alpha^d(S) \triangleq \{ \langle x, y \rangle \mid S \in \mathcal{D}_{\text{diff}} \langle x, y \rangle \}$$

- This is a Galois connection:

**Lemma 1** $\langle \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{\infty}))), \subseteq \rangle \dashv \triangleright \langle L \rightarrow \wp(\mathcal{V} \times \mathcal{V}), \supseteq \rangle$ where the concretization of a dependency property $D \in L \rightarrow \wp(\mathcal{V} \times \mathcal{V})$ is:

$$\gamma^d(D) \triangleq \bigcap_{\ell \in L} \bigcap_{\langle x, y \rangle \in D(\ell)} \mathcal{D}_{\text{diff}} \langle x, y \rangle$$

(the more semantics, the less common dependencies)
Static dependency analysis
Potential dependency

- $\alpha^d(\{S^*[S]\})$ is not computable (Rice theorem)
- We design an over-approximation:

$$\text{Abstract potential dependency semantics } \hat{S}_{\exists}^{\text{diff}} :$$

$$\alpha^d(\{S^{+\infty}[S]\}) \subseteq \hat{S}_{\exists}^{\text{diff}}[S]$$

- The abstraction in D. E. Denning and P. J. Denning, 1977 is purely syntactic;
- We do a little better by taking the semantics is a simple way.
Calculation design

- $\overset{\text{diff}}{\mathcal{S}_d}$ is designed by calculus (in principle can be checked in Coq as Jourdan, Laporte, Blazy, Leroy, and Pichardie, 2015);
- By structural induction on the program syntax;
- By fixpoint approximation for iteration:

**Theorem (fixpoint over-approximation)** If $\langle C, \subseteq, \bot, \top, \lor, \land \rangle$ and $\langle A, \preceq, 0, 1, \lor, \land \rangle$ are complete lattices, $\langle C, \subseteq \rangle \xrightarrow{\gamma} \langle A, \preceq \rangle$ is a Galois connection, $f \in C \rightarrowrightarrow C$ and $\bar{f} \in A \rightarrowrightarrow A$ are monotonically increasing and $\alpha \circ f \preceq \bar{f} \circ \alpha$ (semi-commutation) then $\text{lfp} C f \subseteq \gamma(\text{lfp} A \bar{f})$.

- Finite domain, no need for widening
Abstract potential dependency semantics of assignment $S ::= x = A$;

\[
\begin{align*}
\hat{\mathcal{S}}^\text{diff} [S] \ell &= \begin{cases} 
\{ \ell = \text{at}[S] \? \{ \langle y, y \rangle \mid y \in \mathcal{V} \} 
\end{cases} \\
\{ \ell = \text{after}[S] \? \{ \langle y, x \rangle \mid y \in \hat{\mathcal{S}}^\text{diff} [A] \} \cup \{ \langle y, y \rangle \mid y \neq x \} 
\end{cases} \\
\hat{\mathcal{S}}^\text{diff} [A] &\triangleq \{ y \mid \exists \rho \in \mathcal{E}_v. \exists \nu \in \mathcal{V}. \mathcal{E}[A] \rho \neq \mathcal{E}[A] \rho[y \leftarrow \nu] \}
\end{align*}
\]

\[
\begin{align*}
\hat{\mathcal{S}}^\text{diff} [1] &\triangleq \varnothing \\
\hat{\mathcal{S}}^\text{diff} [x] &\triangleq \{ x \} \\
\hat{\mathcal{S}}^\text{diff} [A_1 - A_2] &\triangleq \{ y \in \text{vars}[A_1] \cup \text{vars}[A_2] \mid A_1 \neq A_2 \}
\end{align*}
\]

\[
\hat{\mathcal{S}}^\text{diff} [A] \subseteq \text{vars}[A]
\]

Examples:

- after $x = y - y$ ; $x$ does not depends on $y$.
- after $x = y$ ; $x = y - x$ ; $x$ depends on the initial value of $x$ and $y$ (to be more precise information of values of variables must be kept such as $y - x = 0$ by symbolic constant analysis)
The case $\ell = \text{at}[S]$ was handled in (44.39). Assume $\ell = \text{after}[S]$.
\[
\alpha^d(\{S^{+\omega}[S]\}) \text{ after}[S] \\
= \alpha^d(\{S^+[S]\}) \text{ after}[S] \tag{def. (7.6) of $S^{+\omega}[S]$ since the assignment $S$ has only finite prefix traces} \\
= \{\langle x', y \rangle \mid S^+[S] \in D_{\text{diff}}(\text{after}[S]) \langle x', y \rangle \} \tag{def. (44.23) of $\alpha^d$ and def. $\subseteq$} \\
= \{\langle x', y \rangle \mid \exists \langle \pi_0, \pi_1, \pi'_0, \pi'_1 \rangle \in S^+[S] . (\forall z \in V \setminus \{x'\} . \varphi(\pi_0)z = \varphi(\pi'_0)z) \wedge \text{diff}(\text{seqval}[y](\text{at}[S])(\pi_0, \pi_1), \text{seqval}[y](\text{at}[S])(\pi'_0, \pi'_1)) \} \tag{def. (44.18) of $D_{\text{diff}}\ell\langle x', y \rangle$} \\
= \{\langle x', y \rangle \mid \exists \langle \pi_0, \pi_1, \pi'_0, \pi'_1 \rangle \in \{\langle \pi_\text{at}[S], \text{at}[S] \rangle \} . (\forall z \in V \setminus \{x'\} . \varphi(\pi_0)z = \varphi(\pi'_0)z) \wedge \text{diff}(\text{seqval}[y](\text{at}[S])(\pi_0, \pi_1), \text{seqval}[y](\text{at}[S])(\pi'_0, \pi'_1)) \} \tag{def. maximal finite trace semantics in Section 6.4 and (6.13)} \\
= \{\langle x', y \rangle \mid \exists \langle \pi_0, \pi_1, \pi'_0, \pi'_1 \rangle \in \{\langle \pi_\text{at}[S], \text{at}[S] \rangle \} . (\forall z \in V \setminus \{x'\} . \varphi(\pi_0)z = \varphi(\pi'_0)z) \wedge \text{diff}(\text{seqval}[y](\text{at}[S])(\pi_0, \pi_1), \text{seqval}[y](\text{at}[S])(\pi'_0, \pi'_1)) \} \tag{def. $\in$} \\
= \{\langle x', y \rangle \mid \exists \langle \pi_0, \pi_1, \pi'_0, \pi'_1 \rangle \in \{\langle \pi_\text{at}[S], \text{at}[S] \rangle \} . (\forall z \in V \setminus \{x'\} . \varphi(\pi_0)z = \varphi(\pi'_0)z) \wedge \text{diff}(\text{seqval}[y](\text{at}[S])(\pi_0, \pi_1), \text{seqval}[y](\text{at}[S])(\pi'_0, \pi'_1)) \} \tag{def. (44.15) of seqval[y]}
Proof II

\[
\subseteq \{\langle x', y \rangle \mid \exists \langle \pi_0 \text{at}[S], \text{at}[S] \rangle \xrightarrow{x = \mathcal{G}[A] Q(\pi_0 \text{at}[S])} \text{after}[S], \langle \pi'_0 \text{at}[S], \text{at}[S] \rangle \xrightarrow{x = \mathcal{G}[A] Q(\pi'_0 \text{at}[S])} \text{after}[S]\} . (\forall z \in \mathcal{V} \setminus \{x'\} . \mathcal{Q}(\pi_0 \text{at}[S])z = \mathcal{Q}(\pi'_0 \text{at}[S])z) \land ((\mathcal{Q}(\pi_0 \text{at}[S])y \neq \mathcal{Q}(\pi'_0 \text{at}[S])y) \lor (\mathcal{Q}(\pi_0 \text{at}[S])y = \mathcal{Q}(\pi'_0 \text{at}[S])y) \land \\
\mathcal{Q}(\pi_0 \text{at}[S]) \xrightarrow{x = \mathcal{G}[A] Q(\pi_0 \text{at}[S])} \text{after}[S]y \neq \mathcal{Q}(\pi'_0 \text{at}[S]) \xrightarrow{x = \mathcal{G}[A] Q(\pi'_0 \text{at}[S])} \text{after}[S]y)\} \quad \text{(44.17) so that } \text{diff}(a \cdot b, c \cdot d) \\
\text{if and only if (1) } a \neq c \text{ or (2) } a = c \land b \neq d.\}
\]

\[
\subseteq \{\langle x', y \rangle \mid \exists \langle \pi_0 \text{at}[S], \text{at}[S] \rangle \xrightarrow{x = \mathcal{G}[A] Q(\pi_0 \text{at}[S])} \text{after}[S], \langle \pi'_0 \text{at}[S], \text{at}[S] \rangle \xrightarrow{x = \mathcal{G}[A] Q(\pi'_0 \text{at}[S])} \text{after}[S]\} . (\forall z \in \mathcal{V} \setminus \{x'\} . \\
\mathcal{Q}(\pi_0 \text{at}[S])z = \mathcal{Q}(\pi'_0 \text{at}[S])z) \land ((y = x') \lor (y = x \land \mathcal{G}[A] \mathcal{Q}(\pi_0 \text{at}[S]) \neq \mathcal{G}[A] \mathcal{Q}(\pi'_0 \text{at}[S])))\}
\]

\[
\subseteq \{\langle x', y \rangle \mid (y = x') \lor (y = x \land \exists \rho, v . \mathcal{G}[A] \rho \neq \mathcal{G}[A] \rho[x' \leftarrow v])\}
\]

\[
\quad \text{\text{letting } } \rho = \mathcal{Q}(\pi_0 \text{at}[S]) \text{ and } v = \mathcal{Q}(\pi'_0 \text{at}[S])(x') \text{ so that } \forall z \in \mathcal{V} \setminus \{x'\} . \mathcal{Q}(\pi_0 \text{at}[S])z = \mathcal{Q}(\pi'_0 \text{at}[S])z \text{ implies that } \mathcal{Q}(\pi'_0 \text{at}[S]) = \rho[x' \leftarrow v]\}
\]

\[
\subseteq \{\langle x', x' \rangle \mid x' \neq x\} \cup \{\langle x', x \rangle \mid \exists \rho, v . \mathcal{G}[A] \rho \neq \mathcal{G}[A] \rho[x' \leftarrow v]\}
\]

\[
= \{\langle x', x' \rangle \mid x' \neq x\} \cup \{\langle x', x \rangle \mid x' \in \mathcal{S}^{\text{diff}}[A]\}
\]

\[
\text{by defining the functional dependency of an expression } A \text{ as } \mathcal{S}^{\text{diff}}[A] \triangleq \{x' \mid \exists \rho, v . \mathcal{G}[A] \rho \neq \mathcal{G}[A] \rho[x' \leftarrow v]\}\]

\[\square\]
Abstract potential dependency semantics of the iteration

\[ S ::= \text{while } \ell (B) S_b \]

\[
\widehat{S}^{\text{diff}} [S] \ell' = (\text{lfp} \frac{\ell}{\mathcal{F}^d [\text{while } \ell (B) S_b]}) \ell'
\]

\[
\mathcal{F}^d [\text{while } \ell (B) S_b] X \ell' =
\]
\[
\{ \ell' = \ell \oplus 1 \lor X(\ell) \lor (X(\ell) \sqcap \widehat{S}^{\text{diff}} [S_b] \ell) \}
\]
\[
\mid \ell' \in \text{in}[S] \lor \{ \text{escape}[S] \oplus \text{break-to}[S] \} : \emptyset \} \lor X(\ell') \lor (X(\ell) \sqcap \widehat{S}^{\text{diff}} [S_b] \ell')
\]
\[
\mid \ell' = \text{after}[S] \oplus X(\ell) \lor \{ (x', y) \mid x' \in \text{vars}[B] \land y \in \text{mod}[S_b] \}
\]
\[
: \emptyset \}
\]

- Can be refined by taking test determinacy into account (e.g. after test \( x == 1 \), \( x \) can only have value 1 so nothing can depend on \( x \) afterwards).

"Calculational design of, a static dependency analysis" – 28/39 – © P. Cousot, NYU, CIMS, CS, April 6th, 2019
No structural compositionality

In the following statement, x and y at $\ell_1$ depend on x at $\ell_0$.

\[\begin{align*}
\ell_0 & \text{ y = x ; } \\
\ell_1 & \text{ /* x = x}_0, y = y_0 */
\end{align*}\]

In the following statement, x and y at $\ell_2$ depend on x at $\ell_1$.

\[\begin{align*}
\ell_1 & \text{ y = y-x ; } \\
\ell_2 & \text{ /* x = x}_0, y = y_0 */
\end{align*}\]

In the sequential composition of the two statements

\[\begin{align*}
\ell_0 & \text{ y = x ; } \\
\ell_1 & \text{ /* x = x}_0, y = x_0 */ \\
\ell_2 & \text{ /* x = x}_0, y = 0 */
\end{align*}\]

y at $\ell_2$ depends on x at $\ell_1$ which depends on x at $\ell_0$ so, by composition, y at $\ell_2$ depends on x at $\ell_0$.

However, y = 0 at $\ell_2$ so y at $\ell_2$ does not depend on x at $\ell_0$. 
Improving precision

- To improve precision one must take values of variables into account;
- Reduced product with a reachability analysis (e.g. Cortesi, Ferrara, Halder, and Zanioli, 2018; Zanioli and Cortesi, 2011)
Conclusion
Dependency analysis is an abstract interpretation

- No need for a generalized theory (as proposed by Assaf, Naumann, Signoles, Totel, and Tronel, 2017; Urban and Müller, 2018)
- This includes further abstractions, dye analysis, taint analysis, etc.
- Many possible variants (e.g. by changing diff to = we get timing channel dependency).
- Data dependency analysis to detect parallelism in sequential codes Padua and Wolfe, 1986 is also an abstract interpretation Tzolovski, 1997, Tzolovski, 2002, Ch. 5.
Bibliographie


References II


References III

References IV


The End, Thank you
Happy sixties Mooly!