Compositional May-Must Program Analysis

Unleashing the Power of Alternation

Patrice Godefroid, Aditya V. Nori, Sriram K. Rajamani
Microsoft Research

Sai Deep Tetali
UC Los Angeles

Property checking

```
void f()
{
0:   *p = 4;
1:   *q = 5;
2:   assert (¬φ_error)
}
```

Question
Does the assertion hold for all possible inputs?

↓

Must analysis: finds bugs, but can't prove their absence
May analysis: can prove the absence of bugs, but can result in false errors

More generally, we are interested in the query

$$\langle \varphi_{pre} \overset{?}{\Rightarrow}_f \varphi_{error} \rangle$$

SMASH = Compositional May-Must Analysis

- *May analysis* = predicate abstraction (SLAM)
- *Must analysis* = symbolic execution + tests (DART)
- *Compositional May-Must analysis*:
  - Interprocedural analysis
  - Memoize and re-use may/must summaries
  - Allows fine-grained coupling and alternation

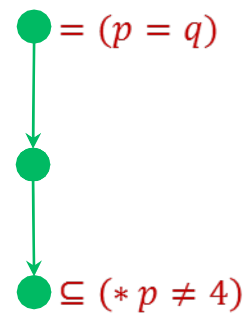
SMASH » Compositional-May || Compositional-Must!

Must information

$\langle T \overset{?}{\Rightarrow}_f (*p \neq 4)\rangle = yes$

```
void f()
{
0:   *p = 4;
1:   *q = 5;
}
```

test



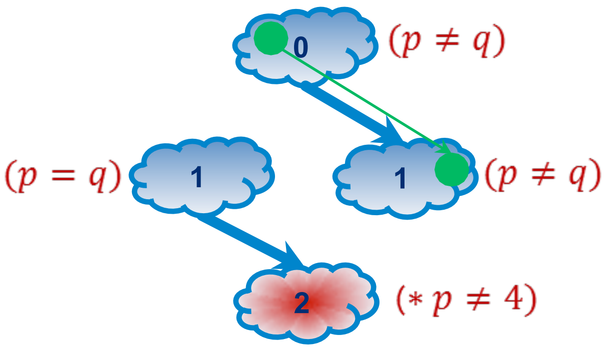
- Captures facts that are guaranteed to hold on particular executions of the program (*under-approximation*)
- Error condition is reachable by any input that satisfies $(p = q)$

May information

$\langle (p \neq q) \overset{?}{\Rightarrow}_f (*p \neq 4) \rangle = no$

```
void f()
{
0:   *p = 4;
1:   *q = 5;
}
```

proof



- Captures facts that are true for all executions of the program (*over-approximation*)
- Proof can be obtained by keeping track of the predicates $(p = q)$ and $(*p \neq 4)$

Must analysis

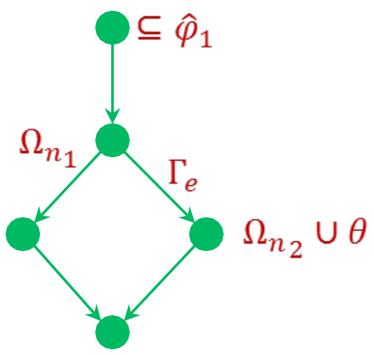
$$\frac{\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle}{\Omega_{n^0_{\mathcal{P}}} := \hat{\varphi}_1 \quad \forall n \in N_{\mathcal{P}} \setminus \{n^0_{\mathcal{P}}\} \,.\, \Omega_n := \emptyset} \text{[INIT} - \text{OMEGA]}$$

● $= \hat{\varphi}_1$

- Associate every program point n with a set of program states $\Omega_n \subseteq \Sigma_{\mathcal{P}}$ (*under-approximation*)
- Initialize Ω_n sets at every program point n: $\Omega_{n^0_{\mathcal{P}}} := \hat{\varphi}_1$

Must analysis

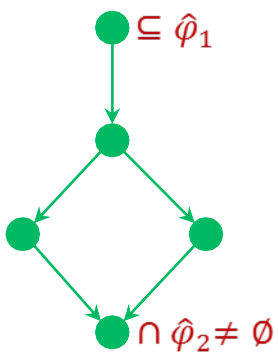
$$\frac{e = (n_1, n_2) \in E_P \quad \theta \subseteq Post(\Gamma_e, \Omega_{n_1})}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST – POST]}$$



- Extend Ω_n sets by forward (under-approximate) analysis
- In particular, use $\theta \subseteq Post(\Gamma_e, \Omega_{n_1})$

Must analysis

$$\frac{\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle \quad \Omega_{n_{\mathcal{P}}^x} \cap \hat{\varphi}_2 \neq \emptyset}{\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle = yes} \text{[BUG – FOUND]}$$



- If an Ω_n state satisfies error condition, $\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle = yes$
- DART [PLDI '05] is a specific instance

May analysis

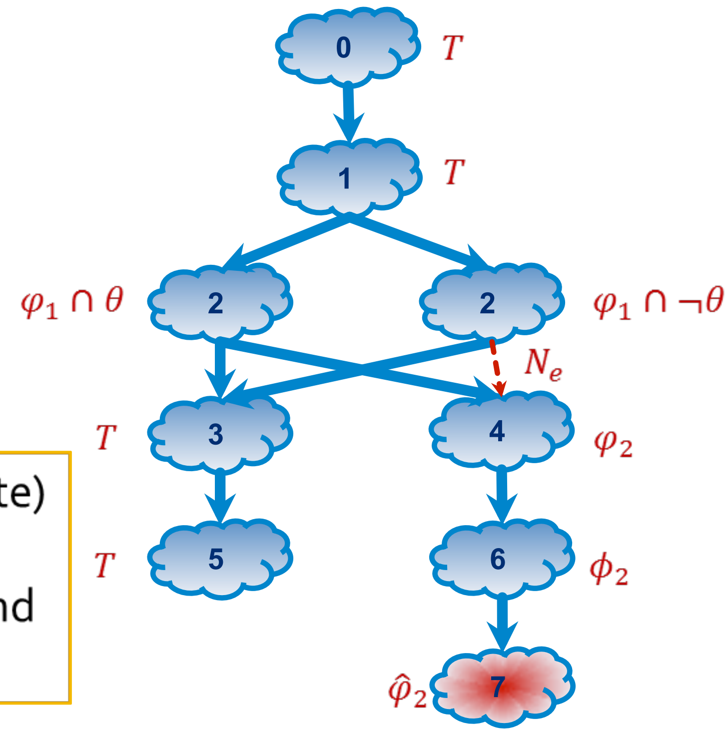
$$\frac{\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle}{\Pi_{n^x_{\mathcal{P}}} := \{\hat{\varphi}_2, \Sigma_{\mathcal{P}} \backslash \hat{\varphi}_2\} \quad \forall n \in N_{\mathcal{P}} \backslash \{n^x_{\mathcal{P}}\}.\ \Pi_n := \{\Sigma_{\mathcal{P}}\} \quad \forall e \in E_{\mathcal{P}}\ .\ N_e := \emptyset} \ [\mathrm{INIT-PI-NE}]$$

- Associate every program point n with a finite partition Π_n of $\Sigma_{\mathcal{P}}$ (*over-approximation*)
- Initialize regions Π_n at every program point n: $\Pi_{n^x_{\mathcal{P}}} := \{\hat{\varphi}_2, \Sigma_{\mathcal{P}} \backslash \hat{\varphi}_2\}$

$\hat{\varphi}_2$

May analysis

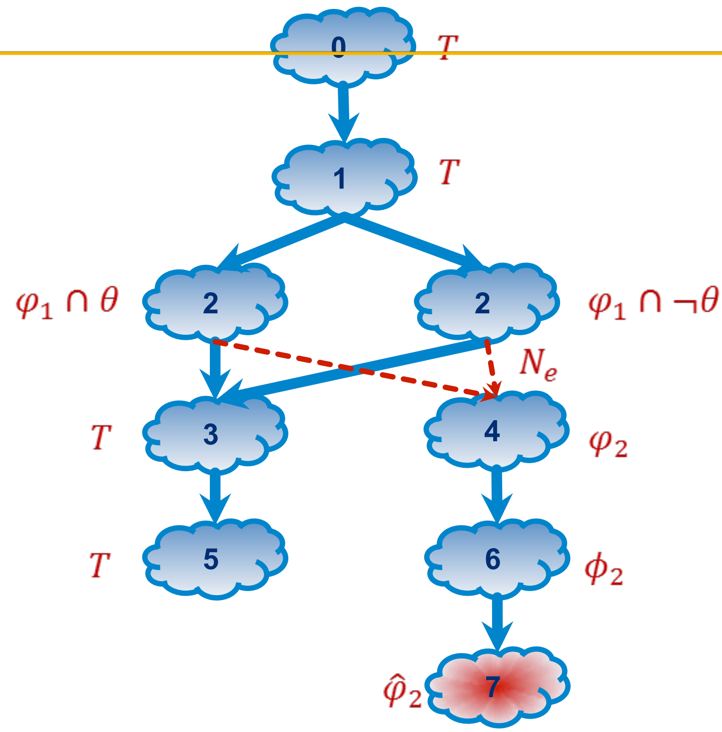
$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad e = (n_1, n_2) \in E_{\mathcal{P}} \quad \theta \supseteq Pre(\Gamma_e, \varphi_2)}{\Pi_{n_1} := \left(\Pi_{n_1} \setminus \{\varphi_1\}\right) \cup \{\varphi_1 \cap \theta, \varphi_1 \cap \neg\theta\} \quad N_e := N_e \cup \{(\varphi_1 \cap \neg\theta, \varphi_2)\}} \text{[NOTMAY} - \text{PRE]}$$



- Refine abstraction via a backward (over-approximate) analysis
- In particular, use $\theta \supseteq Pre(\Gamma_e, \varphi_2)$ for refinement and record deleted abstract edge in N_e

May analysis

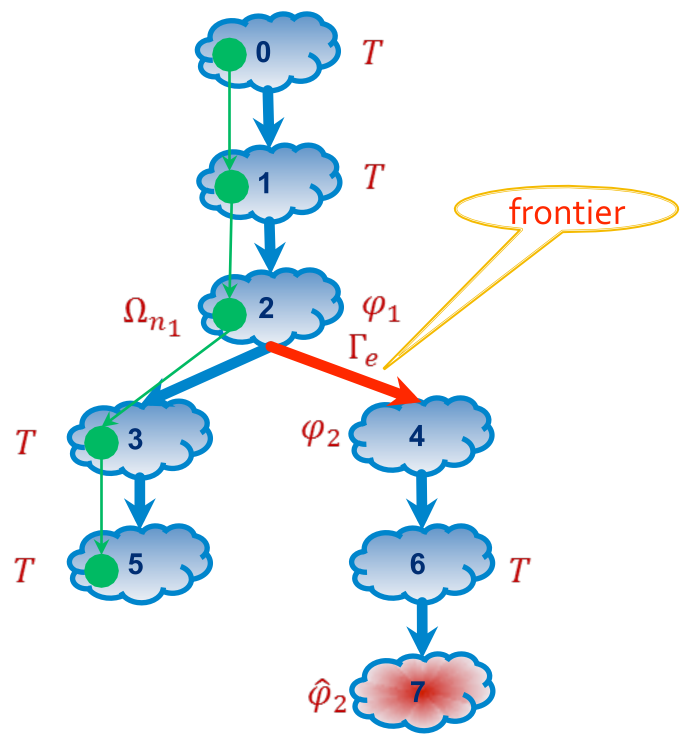
$$\frac{\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle \quad \forall n_0, \dots, n_k \,.\, \forall \varphi_0, \dots, \varphi_k \,.\, n_0 = n_{\mathcal{P}}^0 \wedge n_k = n_{\mathcal{P}}^x \wedge \varphi_0 \in \Pi_{n_0} \wedge \cdots \wedge \varphi_k \in \Pi_{n_k} \wedge \varphi_0 \cap \hat{\varphi}_1 \neq \emptyset \wedge \varphi_k \cap \hat{\varphi}_2 \neq \emptyset \Rightarrow \exists i \in [0, k) \,.\, e = (n_i, n_{i+1}) \in E_{\mathcal{P}} \Rightarrow (\varphi_i, \varphi_{i+1}) \in N_e}{\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle = no} \text{[VERIFIED]}$$



- If the error is unreachable in the abstraction, $\langle \hat{\varphi}_1 \overset{?}{\Rightarrow}_{\mathcal{P}} \hat{\varphi}_2 \rangle = no$
- SLAM [POPL'02] is a specific instance

May-Must analysis

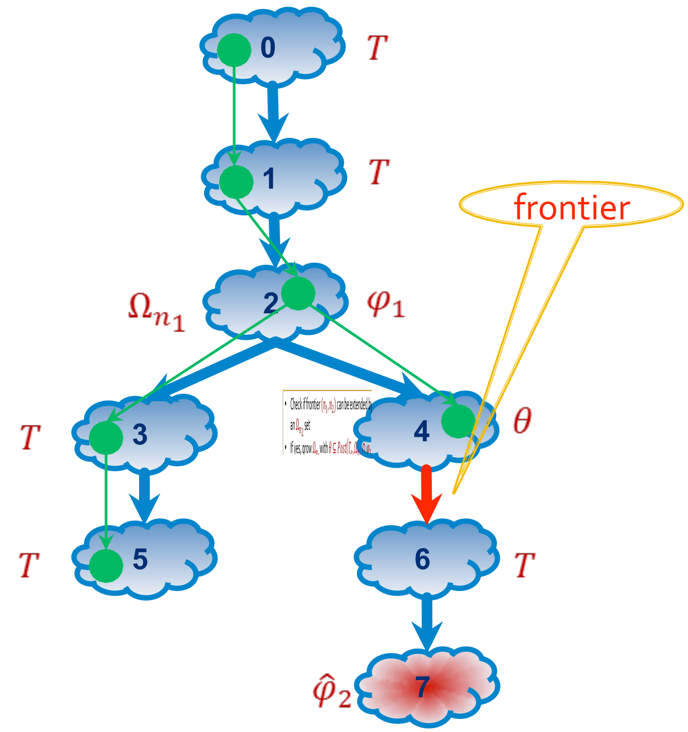
$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad e = (n_1, n_2) \in E_{\mathcal{P}} \qquad \Omega_{n_1} \cap \varphi_1 \neq \emptyset \quad \Omega_{n_2} \cap \varphi_2 = \emptyset \quad \theta \subseteq Post(\Gamma_e, \Omega_{n_1} \cap \varphi_1) \quad \varphi_2 \cap \theta \neq \emptyset}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST} - \text{POST]}$$



- Check if frontier (n_1, n_2) can be extended by an Ω_{n_2} set

May-Must analysis

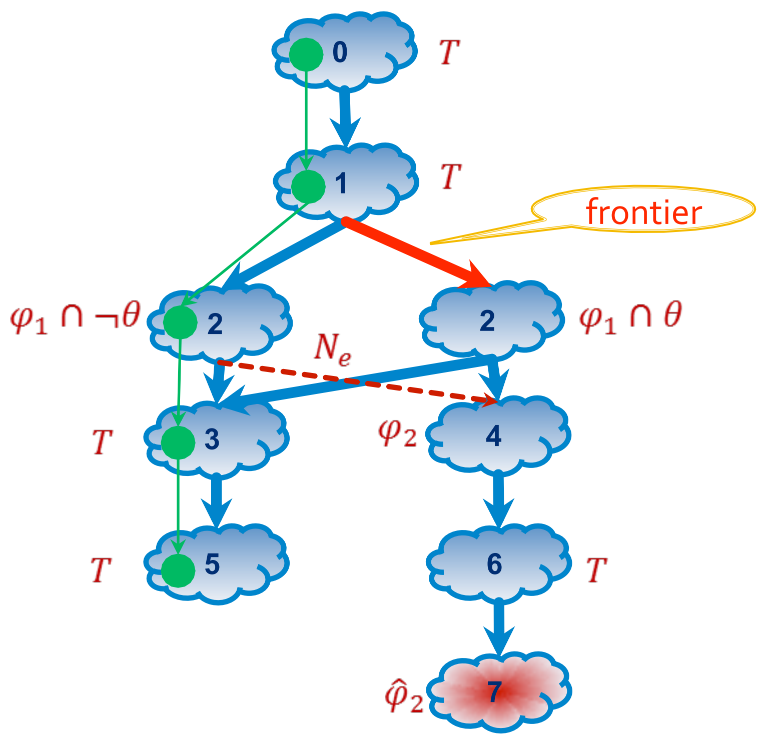
$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad e = (n_1, n_2) \in E_{\mathcal{P}} \qquad \Omega_{n_1} \cap \varphi_1 \neq \emptyset \quad \Omega_{n_2} \cap \varphi_2 = \emptyset \quad \theta \subseteq Post(\Gamma_e, \Omega_{n_1} \cap \varphi_1) \quad \varphi_2 \cap \theta \neq \emptyset}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST – POST]}$$



- Check if frontier (n_1, n_2) can be extended by an Ω_{n_2} set
- If yes, grow Ω_{n_2} with $\theta \subseteq Post(\Gamma_e, \Omega_{n_1} \cap \varphi_1)$

May-Must analysis

$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad e = (n_1, n_2) \in E_{\mathcal{P}} \qquad \Omega_{n_1} \cap \varphi_1 \neq \emptyset \quad \Omega_{n_2} \cap \varphi_2 = \emptyset \quad \theta \supseteq Pre(\Gamma_e, \varphi_2) \quad \theta \cap \Omega_{n_1} = \emptyset}{\Pi_{n_1} := (\Pi_{n_1} \setminus \{\varphi_1\}) \cup \{\varphi_1 \cap \theta, \varphi_1 \cap \neg\theta\} \quad N_e := N_e \cup \{(\varphi_1 \cap \neg\theta, \varphi_2)\}} \text{[NOTMAY − PRE]}$$

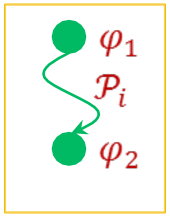


- Check if frontier (n_1, n_2) can be extended by an Ω_{n_2} set
- If not, refine Π_{n_1} with $\theta \supseteq Pre(\Gamma_e, \varphi_2)$ and record deleted abstract edge in N_e
- Synergy/Dash [FSE '06, ISSTA '08] are specific instances

Compositional Must analysis

- A *must summary* for a procedure $\mathcal{P}_i$ is of the form $(\varphi_1, \varphi_2) \in \stackrel{must}{\Longrightarrow}_{\mathcal{P}_i}$
- $\forall t \in \varphi_2 \,.\, \exists s \in \varphi_1 \,.\, t$ can be obtained by executing $\mathcal{P}_i$ from an initial state s

must summary

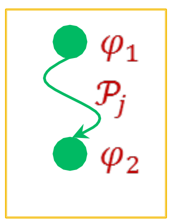


Compositional Must analysis

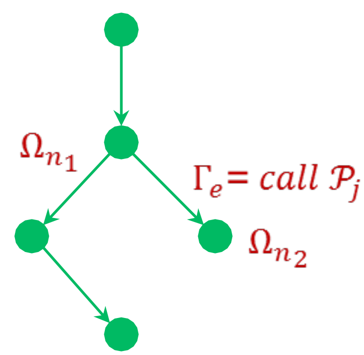
$e = (n_1, n_2) \in E_{\mathcal{P}_i}$ *is a call to procedure* $\mathcal{P}_j$

$$\frac{(\varphi_1, \varphi_2) \in \overset{must}{\Longrightarrow}_{\mathcal{P}_j} \quad \Omega_{n_1} \supseteq \varphi_1 \quad \theta \subseteq \varphi_2}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST – POST – USESUM]}$$

must summary



procedure $\mathcal{P}_i$



- Generate post states by using must summaries

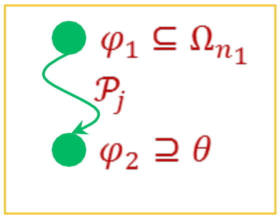
Compositional Must analysis

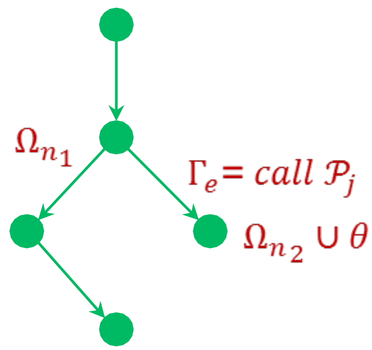
$$e = (n_1, n_2) \in E_{\mathcal{P}_i} \text{ is a call to procedure } \mathcal{P}_j$$

$$\frac{(\varphi_1, \varphi_2) \in \overset{must}{\Longrightarrow}_{\mathcal{P}_j} \quad \Omega_{n_1} \supseteq \varphi_1 \quad \theta \subseteq \varphi_2}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST – POST – USESUM]}$$

procedure $\mathcal{P}_i$

must summary



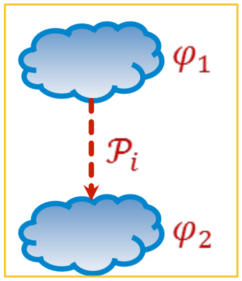


- Generate post states by using *must* summaries
  - ✓ If *must summary*(φ_1, φ_2) is applicable, use $\theta \subseteq \varphi_2$ to extend Ω_{n_2} set
- If no *must* summaries are available for procedure $\mathcal{P}_j$, analyze $\mathcal{P}_j$
- SMART [POPL'07] is a specific instance

Compositional May analysis

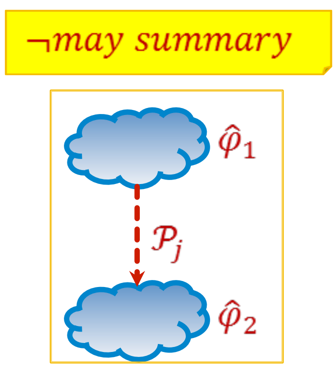
- A $\neg may\ summary$ for a procedure $\mathcal{P}_i$ is of the form $(\varphi_1, \varphi_2) \in \overset{\neg may}{\Longrightarrow}_{\mathcal{P}_i}$
- $\forall s \in \varphi_1\ \forall t \in \varphi_2\ .\ t$ cannot be obtained by executing $\mathcal{P}_i$ starting in state s

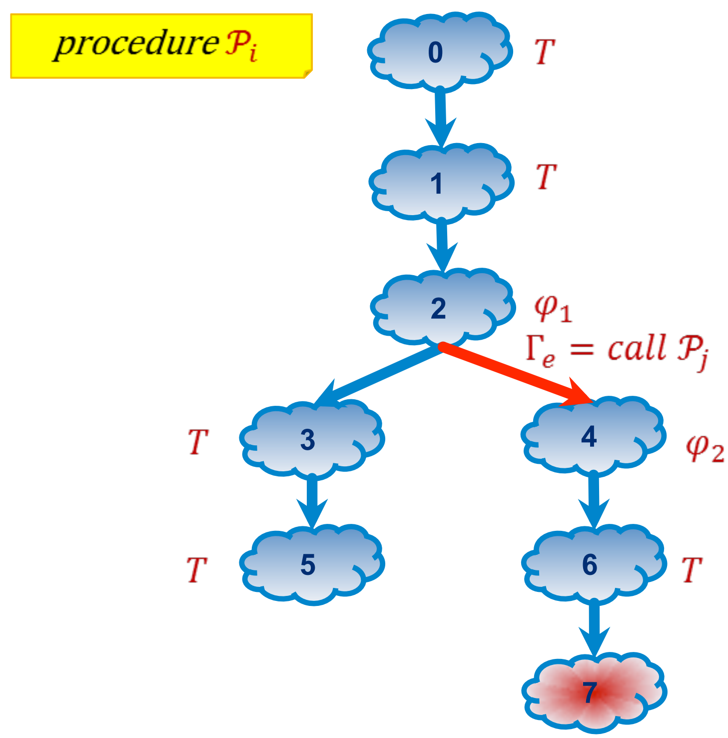
$\neg may\ summary$



Compositional May analysis

$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad e = (n_1, n_2) \in E_{\mathcal{P}_i} \; is\ a\ call\ to\ procedure\ \mathcal{P}_j \qquad (\hat{\varphi}_1, \hat{\varphi}_2) \in \overset{\neg may}{\Longrightarrow}_{\mathcal{P}_j} \quad \varphi_2 \subseteq \hat{\varphi}_2 \quad \theta \subseteq \hat{\varphi}_1}{\Pi_{n_1} := (\Pi_{n_1} \setminus \{\varphi_1\}) \cup \{\varphi_1 \cap \theta, \varphi_1 \cap \neg\theta\} \quad N_e := N_e \cup \{(\varphi_1 \cap \theta, \varphi_2)\}} \; [\text{NMAY} - \text{PRE} - \text{USESUM}]$$

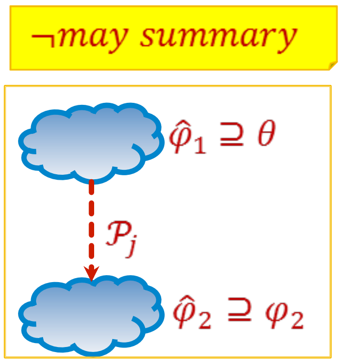


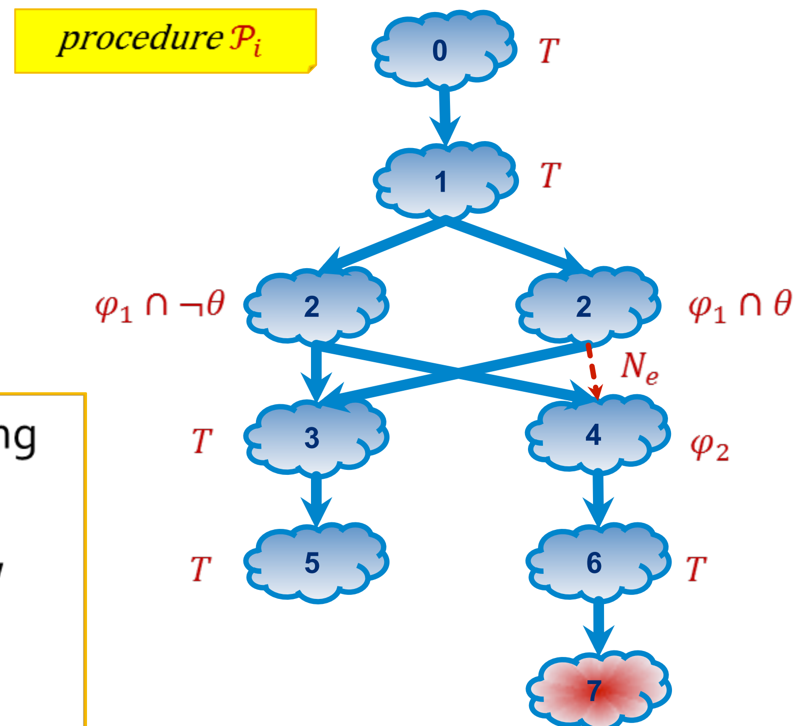


- Refine the abstraction for procedure $\mathcal{P}_i$ by using the $\neg may\ summary$ for $\mathcal{P}_j$

Compositional May analysis

$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad e = (n_1, n_2) \in E_{\mathcal{P}_i} \textit{ is a call to procedure } \mathcal{P}_j \qquad (\hat{\varphi}_1, \hat{\varphi}_2) \in \overset{\neg may}{\Longrightarrow}_{\mathcal{P}_j} \quad \varphi_2 \subseteq \hat{\varphi}_2 \quad \theta \subseteq \hat{\varphi}_1}{\Pi_{n_1} \coloneqq (\Pi_{n_1} \setminus \{\varphi_1\}) \cup \{\varphi_1 \cap \theta, \varphi_1 \cap \neg\theta\} \quad N_e \coloneqq N_e \cup \{(\varphi_1 \cap \theta, \varphi_2)\}} \text{[NMAY} - \text{PRE} - \text{USESUM]}$$



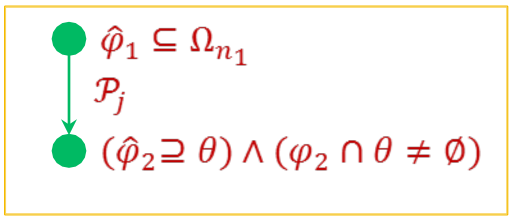


- Refine the abstraction for procedure $\mathcal{P}_i$ by using the $\neg may\ summary$ for $\mathcal{P}_j$
  - ✓ If $\neg may\ summary\ (\hat{\varphi}_1, \hat{\varphi}_2)$ is applicable, use $\theta \subseteq \hat{\varphi}_1$ to refine the abstraction
- If $\neg may$ summaries are not available for procedure $\mathcal{P}_j$, analyze $\mathcal{P}_j$
- SLAM [POPL'02] is a specific instance

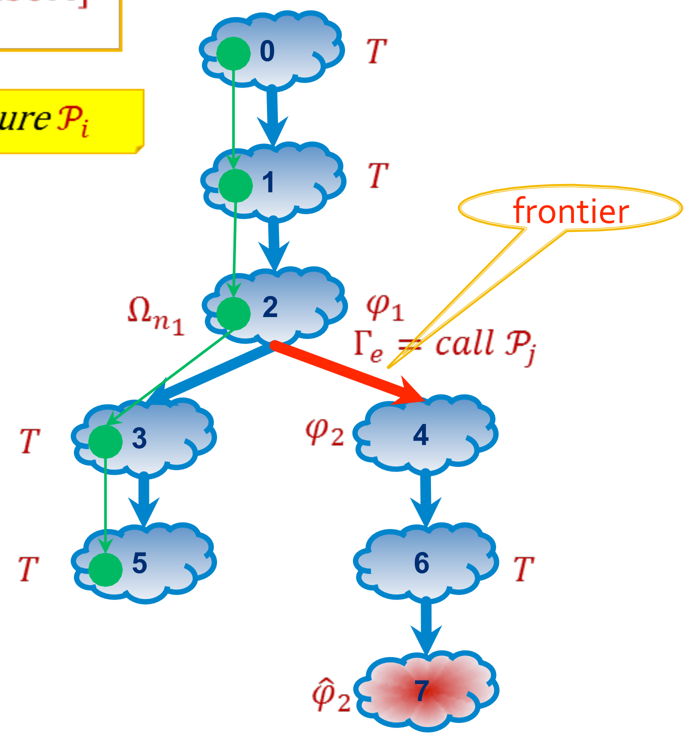
SMASH

$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad \varphi_1 \cap \Omega_{n_1} \neq \emptyset \quad \varphi_2 \cap \Omega_{n_2} = \emptyset \\ e = (n_1, n_2) \in E_{\mathcal{P}_i} \textit{ is a call to procedure } \mathcal{P}_j \\ (\hat{\varphi}_1, \hat{\varphi}_2) \in \overset{must}{\Longrightarrow}_{\mathcal{P}_j} \quad \Omega_{n_1} \supseteq \hat{\varphi}_1 \quad \theta \subseteq \hat{\varphi}_2 \quad \varphi_2 \cap \theta \neq \emptyset}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST} - \text{POST} - \text{USESUM]}$$

must summary



procedure $\mathcal{P}_i$

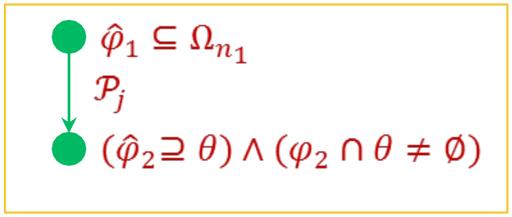


- Base analysis is a may-must analysis (Dash)
- Check if frontier (n_1, n_2) can be extended by a *must summary* $(\hat{\varphi}_1, \hat{\varphi}_2)$

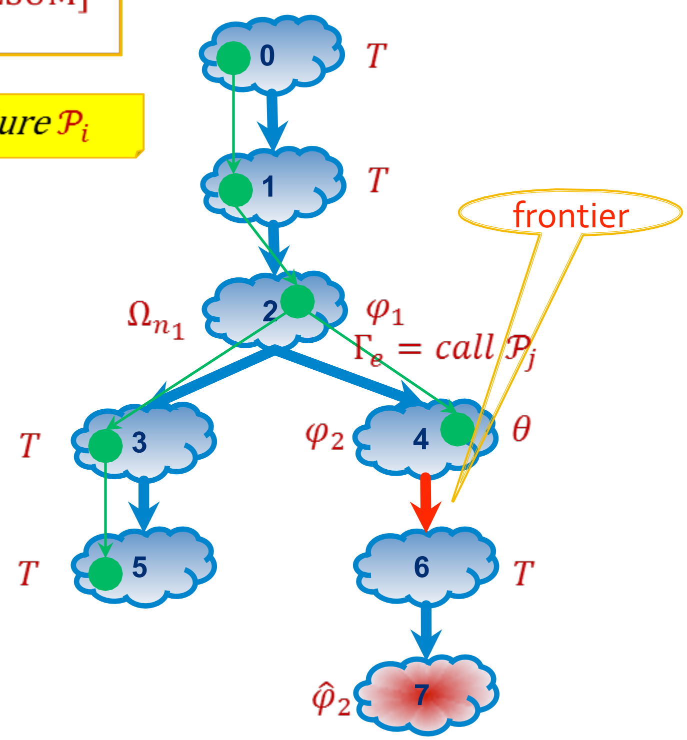
SMASH

$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad \varphi_1 \cap \Omega_{n_1} \neq \emptyset \quad \varphi_2 \cap \Omega_{n_2} = \emptyset \\ e = (n_1, n_2) \in E_{\mathcal{P}_i} \; is\ a\ call\ to\ procedure\ \mathcal{P}_j \\ (\hat{\varphi}_1, \hat{\varphi}_2) \in \overset{must}{\Longrightarrow}_{\mathcal{P}_j} \quad \Omega_{n_1} \supseteq \hat{\varphi}_1 \quad \theta \subseteq \hat{\varphi}_2 \quad \varphi_2 \cap \theta \neq \emptyset}{\Omega_{n_2} := \Omega_{n_2} \cup \theta} \text{[MUST} - \text{POST} - \text{USESUM]}$$

must summary



procedure $\mathcal{P}_i$

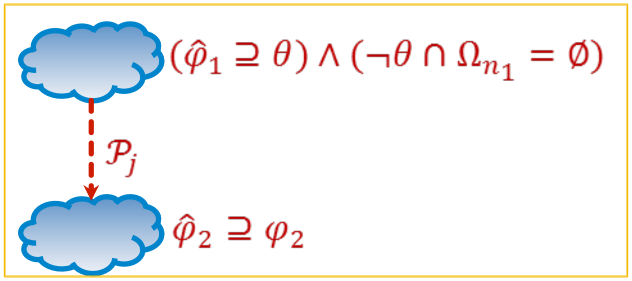


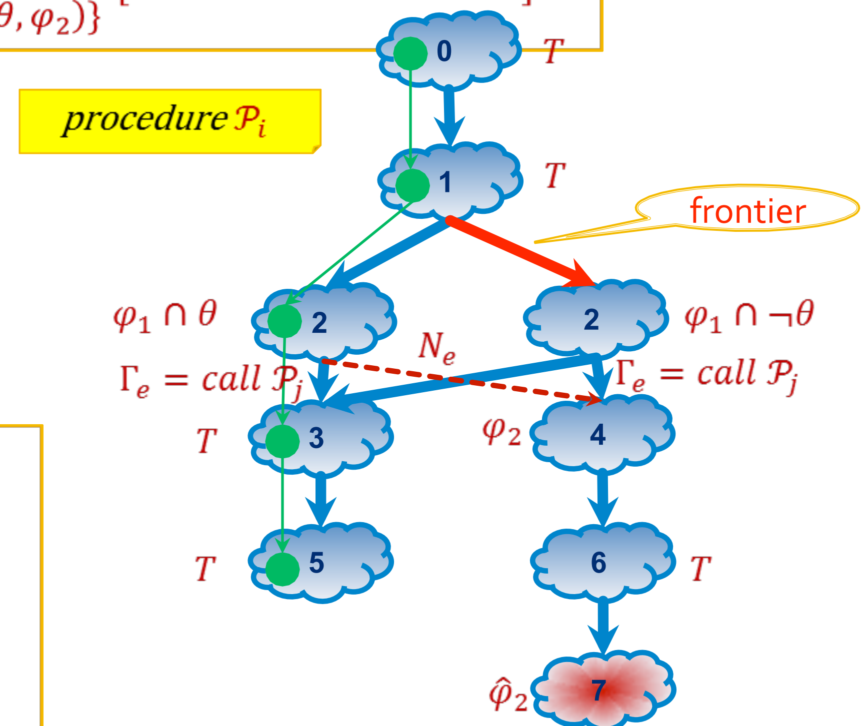
- Check if frontier (n_1, n_2) can be extended by a *must summary* $(\hat{\varphi}_1, \hat{\varphi}_2)$
- If yes, grow Ω_{n_2} with $\theta \subseteq \hat{\varphi}_2$

SMASH

$$\frac{\varphi_1 \in \Pi_{n_1} \quad \varphi_2 \in \Pi_{n_2} \quad \varphi_1 \cap \Omega_{n_1} \neq \emptyset \quad \varphi_2 \cap \Omega_{n_2} = \emptyset \\ e = (n_1, n_2) \in E_{\mathcal{P}_i} \text{ is a call to procedure } \mathcal{P}_j \\ \langle \hat{\varphi}_1, \hat{\varphi}_2 \rangle \in \overset{\neg may}{\Longrightarrow}_{\mathcal{P}_j} \quad \varphi_2 \subseteq \hat{\varphi}_2 \quad \theta \subseteq \hat{\varphi}_1 \quad \neg\theta \cap \Omega_{n_1} = \emptyset}{\Pi_{n_1} := (\Pi_{n_1} \setminus \{\varphi_1\}) \cup \{\varphi_1 \cap \theta, \varphi_1 \cap \neg\theta\} \quad N_e := N_e \cup \{(\varphi_1 \cap \theta, \varphi_2)\}} \text{[NMAY – PRE – USESUM]}$$

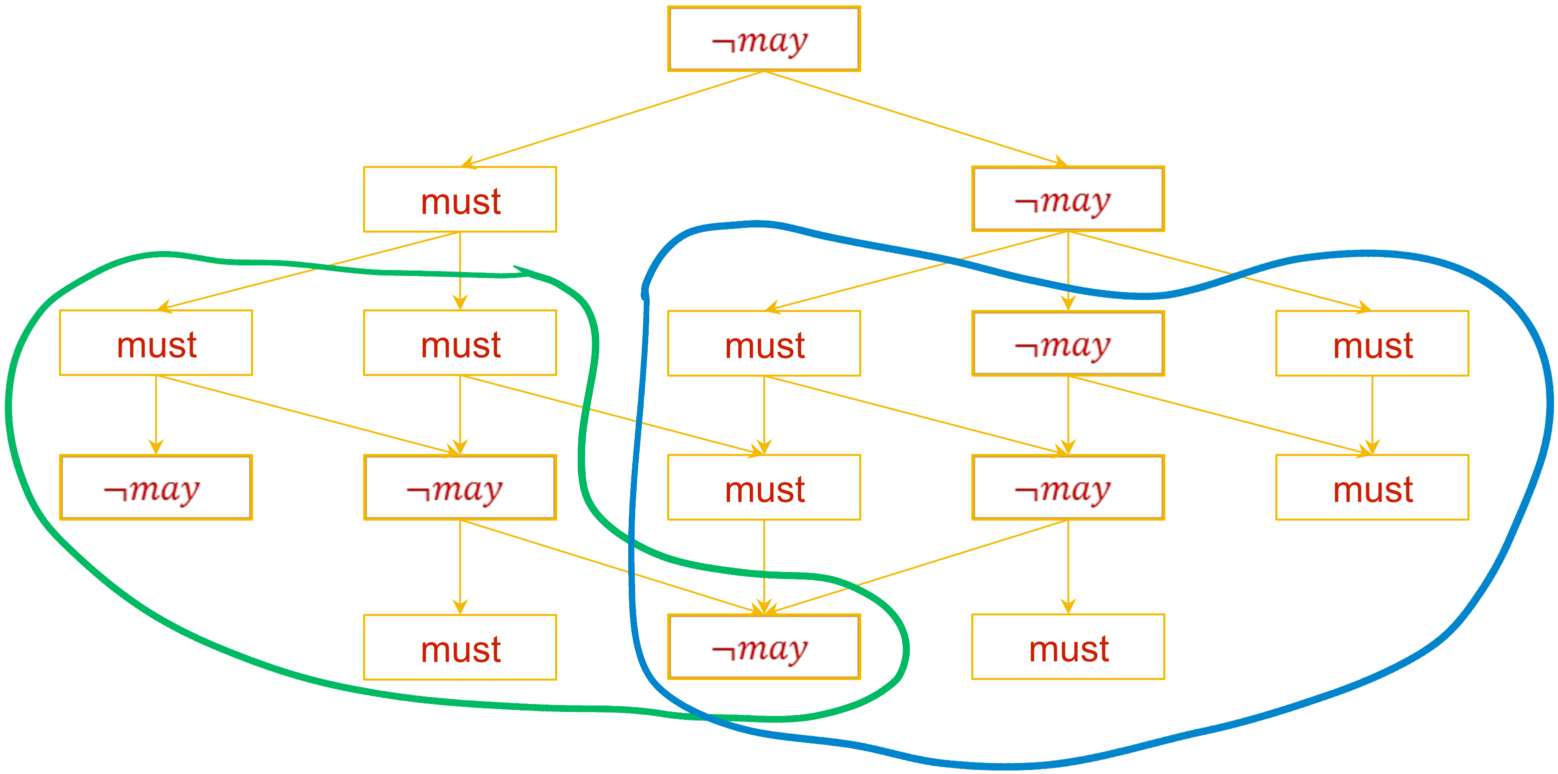
$\neg may\ summary$





- Check if frontier (n_1, n_2) can be refined by a $\neg may\ summary\ (\hat{\varphi}_1, \hat{\varphi}_2)$
- If yes, use $\theta \subseteq \hat{\varphi}_1$ to refine the abstraction
- If both $must$ and $\neg may$ summaries are not available, analyze procedure $\mathcal{P}_j$
  - $yes \Rightarrow must\ summary$ for $\mathcal{P}_j$
  - $no \Rightarrow \neg may\ summary$ for $\mathcal{P}_j$

Interplay between $\neg may$ and $must$ summaries



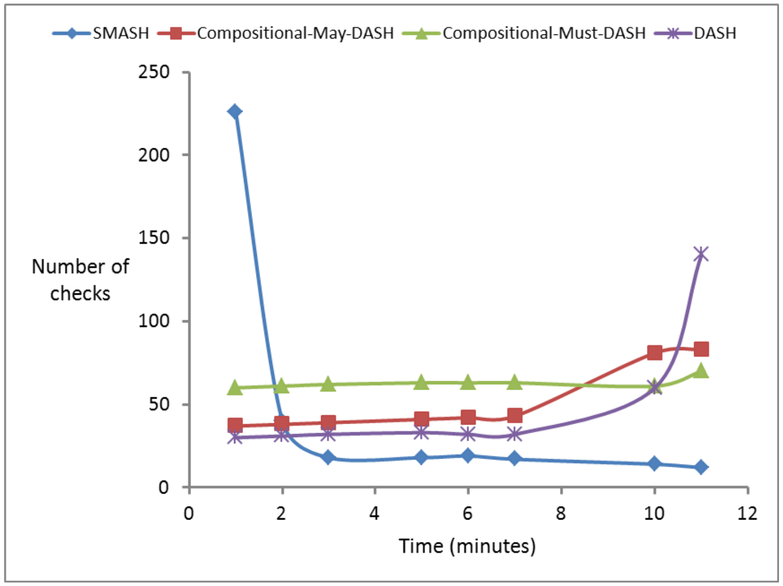
Implementation

- The SMASH implementation is a deterministic realization of the declarative rules
- Input C program is first abstractly interpreted
  - No pointer arithmetic -- *(p+i) is treated as *p
  - Logic encoding -- propositional logic, linear arithmetic and uninterpreted functions
- Theorem prover: Z3

Evaluation on Windows 7 drivers

Statistics	Dash	SMASH
Average $\neg$*may summaries*/driver	0	39
Average must *summaries*/driver	0	12
Number of proofs	2176	2228
Number of bugs	64	64
Time-outs	61	9
Time (hours)	117	44

69 drivers (342000 LOC) and 85 properties



We have unleashed the power of alternation!

Summary

- SMASH is a unified framework for compositional may-must program analysis
- We have explained SMASH in the context of existing analyses (SLAM, DART, Synergy/Dash ...) in the area
- Empirical evaluation shows that SMASH can significantly outperform may-only, must-only and non-compositional may-must algorithms

http://research.microsoft.com/yogi