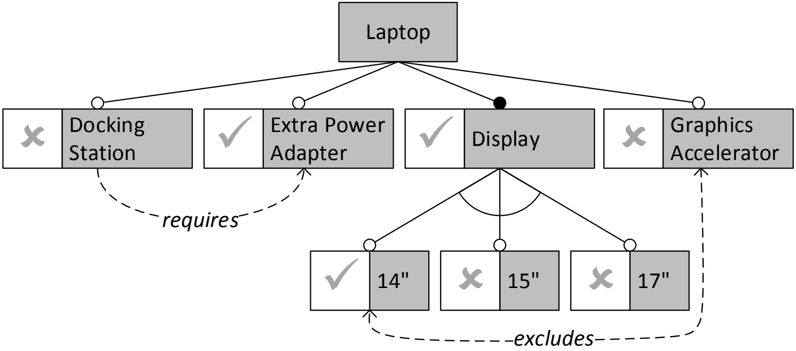
On Lazy and Eager Interactive Reconfiguration

Mikoláš Janota[1]
Goetz Botterweck[2] Joao Marques-Silva[1,3]

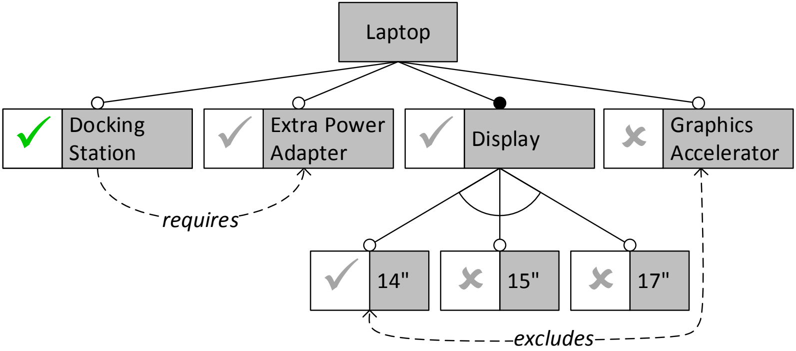
[1] INESC-ID/IST, Lisbon, Portugal
[2]Lero, Limerick, Ireland
[3] CASL/CSI, University College Dublin, Ireland

VAMOS 2014, January 22–24
Nice, France

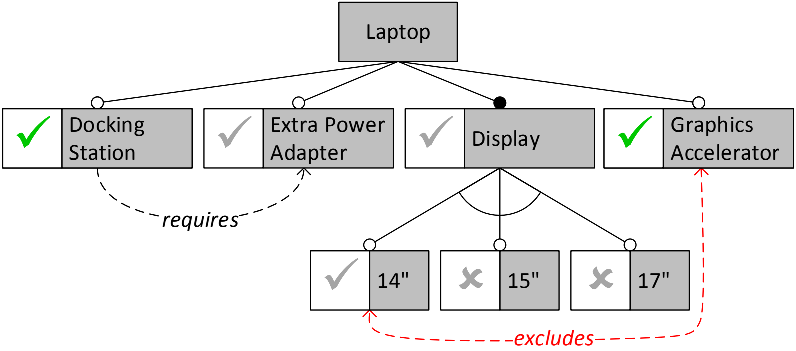
Reconfiguration Example



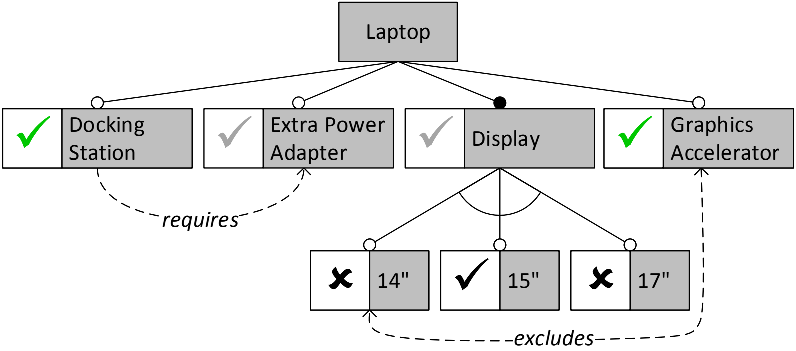
Reconfiguration Example



Reconfiguration Example



Reconfiguration Example



Scenarios in Reconfiguration

- LEGAL: can be changed without further changes

Scenarios in Reconfiguration

- LEGAL: can be changed without further changes
- RECONFIGURE: can be changed but some other default decisions need to be changed.

Scenarios in Reconfiguration

- LEGAL: can be changed without further changes
- RECONFIGURE: can be changed but some other default decisions need to be changed.
- ILLEGAL: cannot be changed without violating the formula or without altering other user decisions.

Some Concepts from (Propositional) Logic

- CNF: express an instance *clauses*

Some Concepts from (Propositional) Logic

- CNF: express an instance *clauses*
  - E.g.

$$\{(\neg x \vee z), (\neg y \vee z), (\neg x \vee \neg y)\}$$

Some Concepts from (Propositional) Logic

- CNF: express an instance *clauses*
  - E.g.

$$\{(\neg x \vee z), (\neg y \vee z), (\neg x \vee \neg y)\}$$

- A CNF is *satisfiable* if there's a truth assignment that makes all the clauses true. It is *unsatisfiable* otherwise. We can use a SAT solver to decide formula's satisfiability.

Some Concepts from (Propositional) Logic

- CNF: express an instance *clauses*
  - E.g.

$$\{(\neg x \vee z), (\neg y \vee z), (\neg x \vee \neg y)\}$$

- A CNF is *satisfiable* if there's a truth assignment that makes all the clauses true. It is *unsatisfiable* otherwise. We can use a SAT solver to decide formula's satisfiability.
- A *Minimally Unsatisfiable Set (MUS)* of clauses is a irreducible unsatisfiable CNF.

Some Concepts from (Propositional) Logic

- CNF: express an instance *clauses*
  - E.g.

$$\{(\neg x \vee z), (\neg y \vee z), (\neg x \vee \neg y)\}$$

- A CNF is *satisfiable* if there's a truth assignment that makes all the clauses true. It is *unsatisfiable* otherwise. We can use a SAT solver to decide formula's satisfiability.
- A *Minimally Unsatisfiable Set (MUS)* of clauses is a irreducible unsatisfiable CNF.
- A *Minimal Correction Subset (MCS)* of clauses is a irreducible set of clauses whose removal makes the original formula satisfiable.

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance
- $\mathcal{D}^a$ ... decisions made by the agent

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance
- $\mathcal{D}^a$ ... decisions made by the agent
  - E.g. $\{\neg x, y\}$

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance
- $\mathcal{D}^a$ ... decisions made by the agent
  - E.g. $\{\neg x, y\}$
- $\mathcal{D}^d$ ... default decisions.

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance
- $\mathcal{D}^a$ ... decisions made by the agent
  - E.g. $\{\neg x, y\}$
- $\mathcal{D}^d$ ... default decisions.

Changing from decision l to $\neg l$

- LEGAL: Just update $\mathcal{D}^a$ and $\mathcal{D}^d$

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance
- $\mathcal{D}^a$ ... decisions made by the agent
  - E.g. $\{\neg x, y\}$
- $\mathcal{D}^d$ ... default decisions.

Changing from decision l to $\neg l$

- LEGAL: Just update $\mathcal{D}^a$ and $\mathcal{D}^d$
- RECONFIGURE: Change $\mathcal{D}^d$ so that the new decision can be accommodated. Use MCSes for that.

Handling Reconfiguration Scenarios

Modeling the problem.

- $\mathcal{F}$ ... configuration instance
- $\mathcal{D}^a$ ... decisions made by the agent
  - E.g. $\{\neg x, y\}$
- $\mathcal{D}^d$ ... default decisions.

Changing from decision l to $\neg l$

- LEGAL: Just update $\mathcal{D}^a$ and $\mathcal{D}^d$
- RECONFIGURE: Change $\mathcal{D}^d$ so that the new decision can be accommodated. Use MCSes for that.
- ILLEGAL: Provide an explanation why.

Lazy Approach

Function $\texttt{Change}$ (l)

begin

$(\texttt{st}_1, \mu_1, \texttt{core}_1) \leftarrow \texttt{SAT}(\mathcal{F} \cup \mathcal{D}^a \cup \mathcal{D}^d \setminus \{l\} \cup \{\neg l\})$

$(\texttt{st}_2, \mu_1, \texttt{core}_2) \leftarrow \texttt{SAT}(\mathcal{F} \cup \mathcal{D}^a \setminus \{l\} \cup \{\neg l\})$

if $\texttt{st}_1$ **then**

  $\mathcal{D}^a \leftarrow \mathcal{D}^a \setminus \{l\} \cup \{\neg l\}$

  $\mathcal{D}^d \leftarrow \mathcal{D}^d \setminus \{l\}$

  return LEGAL

if $\texttt{st}_2$ **then**

  $\mathcal{D}^a \leftarrow \mathcal{D}^a \setminus \{l\} \cup \{\neg l\}$

  $\mathcal{D}^d \leftarrow \mathcal{D}^d \setminus \{l\}$

  $\mathcal{D}^d = \texttt{Reconfigure}(\mathcal{F} \wedge \mathcal{D}^a, \mathcal{D}^d)$

  return RECONFIGURE

$\texttt{Explain}(\mathcal{F}, \mathcal{D}^a \setminus \{l\} \cup \{\neg l\})$

return ILLEGAL

Eager Approach—Change

If $\mathcal{F} \cup \mathcal{D}^a \cup \mathcal{D}^d \smallsetminus \{l\} \models l$, remember a reason $\mathtt{R}_l \subseteq \mathcal{D}^a \cup \mathcal{D}^d$ s.t. $\mathcal{F} \cup \mathtt{R}_l \models l$.

Function `Change` (l)
begin
    foreach k *s.t.* $l \in \mathtt{R}_k$ **do**
        remove $\mathtt{R}_k$ as reason for k
        mark k as LEGAL
    foreach k *is* LEGAL **do**
        `Check`(k)

Eager Approach—Check Literal

Function $\texttt{Check}\ (l)$
begin
    $(\texttt{st}_1, \mu_1, \texttt{core}_1) \leftarrow \texttt{SAT}(\mathcal{F} \cup \mathcal{D}^a \cup \mathcal{D}^d \setminus \{l\} \cup \{\neg l\})$
    $(\texttt{st}_2, \mu_1, \texttt{core}_2) \leftarrow \texttt{SAT}(\mathcal{F} \cup \mathcal{D}^a \setminus \{l\} \cup \{\neg l\})$
    if $\texttt{st}_1$ **then**
        mark l as LEGAL
    else if $\texttt{st}_2$ **then**
        mark l as RECONFIGURE
        $\texttt{R}_l \leftarrow \texttt{core}_1 \cap (\mathcal{D}^a \cup \mathcal{D}^d) \setminus \{\neg l\}$
    else
        mark l as ILLEGAL
        $\texttt{R}_l \leftarrow \texttt{core}_2 \cap \mathcal{D}^a \setminus \{\neg l\}$

Experimental Results

	SPLOT rl	SPLOT rnd	LVAT mod	LVAT hard
Algorithm	eager	eager	lazy	lazy
Maximal time	0.008 s	1.3 s	2 s	41.76 s
time < 0.5 s	100%	99.8%	99.9%	86%
time < 1.5 s	100%	100%	99.9%	91%
# RECONF	4199	79860	43821	28259
# ILLEGAL	3987	97103	167795	55903
# LEGAL	4314	73037	33384	5838
Max. RECON	0.006 s	0.5 s	2 s	41.76 s
Max. ILLEG	0.001 s	0.02 s	0.92 s	11.10 s
Max. LEGAL	0.006 s	0.32 s	0.005 s	0.01 s
Max. INITIAL	0.008 s	1.3 s	0.01 s	0.15 s
# models	25	10	10	4
# variables	60–366	10,000–10,000	684–14910	23,516–62,482

Conclusions and Future Work

- SAT-based support for reconfiguration.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.
- Developed a *lazy algorithm*, which computes a status of a variable on demand.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.
- Developed a *lazy algorithm*, which computes a status of a variable on demand.
- Developed an *eager algorithm*, which maintains a status of each variable at all times.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.
- Developed a *lazy algorithm*, which computes a status of a variable on demand.
- Developed an *eager algorithm*, which maintains a status of each variable at all times.
- Eager—10^2 of variables, Lazy—10^4 variables.

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.
- Developed a *lazy algorithm*, which computes a status of a variable on demand.
- Developed an *eager algorithm*, which maintains a status of each variable at all times.
- Eager—10^2 of variables, Lazy—10^4 variables.
- How to deal with "large" reconfiguration?

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.
- Developed a *lazy algorithm*, which computes a status of a variable on demand.
- Developed an *eager algorithm*, which maintains a status of each variable at all times.
- Eager—10^2 of variables, Lazy—10^4 variables.
- How to deal with "large" reconfiguration?
- How to apply reconfiguration in the vicinity only?

Conclusions and Future Work

- SAT-based support for reconfiguration.
- 3 types of scenarios: LEGAL, RECONFIGURE, and ILLEGAL.
- For RECONFIGURE, use MCS to change other variables.
- For ILLEGAL, use MUS to explain.
- Developed a *lazy algorithm*, which computes a status of a variable on demand.
- Developed an *eager algorithm*, which maintains a status of each variable at all times.
- Eager—10^2 of variables, Lazy—10^4 variables.
- How to deal with "large" reconfiguration?
- How to apply reconfiguration in the vicinity only?
- What do users want from reconfiguration?

Thank you for your attention!

Questions?