Computing Minimal Unsatisfiable Subsets of Constraints

Author: Shahriar Robbani
Supervision: Erika Ábrahám

Theory of Hybrid Systems - Informatik 2 - RWTH-Aachen

Seminar Winter-16/17

Outline

Fundamentals

- **Propositional Logic Formula:** A well-formed propositional logic has following grammar:

$$\varphi \quad := \quad a \mid (\neg\varphi) \mid (\varphi \wedge \varphi)$$

- **Literals:** A literal is a positive or negative instance of Boolean variable. For example, x or $\neg x$.
- **Clause:** It is a disjunction of literals. For example, $C = (a \vee \neg b \vee c)$.
- **Conjunctive Normal Form (CNF):** A CNF formula φ is defined as follows:

$$\varphi = \bigwedge_{i=1\ldots n} C_i$$

- **Clause-Selector Variable:** A clause-selector variable, w_i is defined as:

$$C'_i = (\neg w_i \vee C_i)$$

Minimal Unsatisfiable Subsets and Minimal Correction Subset

Minimal Unsatisfiable Subset (MUSs):

(x)	$(\neg x)$	$(\neg x \vee y)$	$(\neg x \vee \neg y)$
$\square$	$\square$		
$\square$		$\square$	

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Minimal Unsatisfiable Subsets and Minimal Correction Subset

Minimal Unsatisfiable Subset (MUSs):

(x)	$(\neg x)$	$(\neg x \vee y)$	$(\neg x \vee \neg y)$
$\square$	$\square$		
$\square$		$\square$	

Minimal Unsatisfiable Subset (MUSs):

(x)	$(\neg x)$	$(\neg x \vee y)$	$(\neg x \vee \neg y)$
✓			
	✓	✓	
	✓		✓

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

MUS \ MCS Duality

- **Hitting Sets:**

  $D = \{a, b, c, d\}$
  $\Omega = \{(a,b),(b,c,d)\}$
  $H = \{(a,b), (b,c), b, \ldots\}$

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, . . .}

MUS \ MCS Duality

- **Hitting Sets:**

  $D = \{a, b, c, d\}$

  $\Omega = \{(a,b),(b,c,d)\}$

  $H = \{(a,b), (b,c), b, \ldots\}$

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}

  Ω = {(a,b),(b,c,d)}

  H = {(a,b), (b,c), b, . . .}

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, . . .}
  MinH = {(a,c), (a,d), b}

MUS \ MCS Duality

- **Hitting Sets:**

  $D = \{a, b, c, d\}$
  $\Omega = \{(a,b),(b,c,d)\}$
  $H = \{(a,b), (b,c), b, \ldots\}$
  $MinH = \{(a,c), (a,d), b\}$

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, . . .}
  MinH = {(a,c), (a,d), b}

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, . . .}
  MinH = {(a,c), (a,d), b}

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, . . .}
  MinH = {(a,c), (a,d), b}

- The set of MUSs of a formula φ is equal to the set of minimal hitting sets of the set of MCSs.

$$MCS_1 = \{C_1\}$$
$$MCS_2 = \{C_2, C_3\}$$
$$MCS_3 = \{C_2, C_4\}$$

$$MUS_1 = \{C_1, C_2\}$$
$$MUS_2 = \{C_1, C_3, C_4\}$$

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, ...}
  MinH = {(a,c), (a,d), b}

- The set of MUSs of a formula φ is equal to the set of minimal hitting sets of the set of MCSs.

$$
\begin{aligned}
MCS_1 &= \{C_1\} \\
MCS_2 &= \{C_2, C_3\} \\
MCS_3 &= \{C_2, C_4\}
\end{aligned}
$$

$$
\begin{aligned}
MUS_1 &= \{C_1, C_2\} \\
MUS_2 &= \{C_1, C_3, C_4\}
\end{aligned}
$$

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

MUS \ MCS Duality

- **Hitting Sets:**

  D = {a, b, c, d}
  Ω = {(a,b),(b,c,d)}
  H = {(a,b), (b,c), b, . . .}
  MinH = {(a,c), (a,d), b}

- The set of MUSs of a formula φ is equal to the set of minimal hitting sets of the set of MCSs.

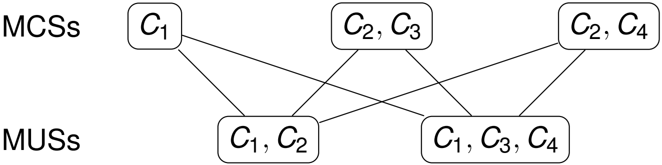
$$MCS_1 = \{C_1\}$$
$$MCS_2 = \{C_2, C_3\}$$
$$MCS_3 = \{C_2, C_4\}$$

$$MUS_1 = \{C_1, C_2\}$$
$$MUS_2 = \{C_1, C_3, C_4\}$$

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

MUS \ MCS Duality (cnt...)

- Additionally, each MCS is an minimal hitting set of the set of MUSs.
- So, minimal hitting sets of MUSs and MCSs provide a transformation from one collection to the other. This is the duality of MUS and MCS.



$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Approach

1. Computing all MCSs
2. Computing Hitting Sets of MCSs

Algorithm: Computing all MCSs

- Augment CNF with clause selector variables

$$\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

$$\Downarrow$$

$$\varphi' = (\neg w_1 \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

Algorithm: Computing all MCSs

- Augment CNF with clause selector variables

$$\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

$$\Downarrow$$

$$\varphi' = (\neg w_1 \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

- Find a solution to the augmented formula with the fewest *w*-variables assigned **false**

$$\varphi' = (\neg \textbf{false} \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

Algorithm: Computing all MCSs

- Augment CNF with clause selector variables

$$\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

$$\Downarrow$$

$$\varphi' = (\neg w_1 \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

- Find a solution to the augmented formula with the fewest *w*-variables assigned **false**

$$\varphi' = (\neg \textbf{false} \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

- Add blocking clauses to block old solutions

$$\varphi' = \varphi' \wedge w_1$$

Algorithm: Computing all MCSs

- Augment CNF with clause selector variables

$$\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

$$\Downarrow$$

$$\varphi' = (\neg w_1 \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

- Find a solution to the augmented formula with the fewest *w*-variables assigned **false**

$$\varphi' = (\neg \textbf{false} \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

- Add blocking clauses to block old solutions

$$\varphi' = \varphi' \wedge w_1$$

- Find MCSs incrementally until all are found.

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Clauses

1. x
2. $\neg x$
3. $\neg x \vee y$
4. $\neg x \vee \neg y$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \land \underbrace{(\neg x)}_{C_2} \land \underbrace{(\neg x \lor y)}_{C_3} \land \underbrace{(\neg x \lor \neg y)}_{C_4}$$

1. Add clause-selector variables.

Clauses

1. $\neg w_1 \lor x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \lor y$
4. $\neg w_4 \neg x \lor \neg y$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

1. Add clause-selector variables.
2. Add AtMost constraint.

Clauses

1. $\neg w_1 \vee x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \vee y$
4. $\neg w_4 \neg x \vee \neg y$

$$AtMost(\{w_1, w_2, w_3, w_4\}, 1)$$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

1. Add clause-selector variables.
2. Add AtMost constraint.
3. First solution : w1 is **false**. Add blocking clause and a MCS.

Clauses

1. $\neg w_1 \vee x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \vee y$
4. $\neg w_4 \neg x \vee \neg y$
5. w_1

MCSs

1. $\{x\}$

$$AtMost(\{w_1, w_2, w_3, w_4\}, 1)$$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Clauses

1. $\neg w_1 \vee x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \vee y$
4. $\neg w_4 \neg x \vee \neg y$
5. w_1

1. Add clause-selector variables.
2. Add AtMost constraint.
3. First solution : w1 is **false**. Add blocking clause and a MCS.
4. No further solutions, increment AtMost.

MCSs

1. $\{x\}$

$$AtMost(\{w_1, w_2, w_3, w_4\}, 2)$$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Clauses

1. $\neg w_1 \vee x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \vee y$
4. $\neg w_4 \neg x \vee \neg y$
5. w_1
6. $w_2 \vee w_3$

1. Add clause-selector variables.
2. Add AtMost constraint.
3. First solution : w1 is **false**. Add blocking clause and a MCS.
4. No further solutions, increment AtMost.
5. Second solution : w2 and w3 are **false**. Add blocking clause and another MCSs.

MCSs

1. $\{x\}$
2. $\{\neg x, \neg x \vee y\}$

$$AtMost(\{w_1, w_2, w_3, w_4\}, 2)$$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Clauses

1. $\neg w_1 \vee x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \vee y$
4. $\neg w_4 \neg x \vee \neg y$
5. w_1
6. $w_2 \vee w_3$
7. $w_2 \vee w_4$

---

1. Add clause-selector variables.
2. Add AtMost constraint.
3. First solution : w1 is **false**. Add blocking clause and a MCS.
4. No further solutions, increment AtMost.
5. Second solution : w2 and w3 are **false**. Add blocking clause and another MCSs.
6. Third solution : w2 and w4 are **false**. Add blocking clause and another MCSs.

MCSs

1. $\{x\}$
2. $\{\neg x, \neg x \vee y\}$
3. $\{\neg x, \neg x \vee \neg y\}$

$$AtMost(\{w_1, w_2, w_3, w_4\}, 2)$$

Example: Computing all MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Clauses

1. $\neg w_1 \vee x$
2. $\neg w_2 \neg x$
3. $\neg w_3 \neg x \vee y$
4. $\neg w_4 \neg x \vee \neg y$
5. w_1
6. $w_2 \vee w_3$
7. $w_2 \vee w_4$

1. Add clause-selector variables.
2. Add AtMost constraint.
3. First solution : w1 is **false**. Add blocking clause and a MCS.
4. No further solutions, increment AtMost.
5. Second solution : w2 and w3 are **false**. Add blocking clause and another MCSs.
6. Third solution : w2 and w4 are **false**. Add blocking clause and another MCSs.
7. No further solutions, even without AtMost constraint.

MCSs

1. $\{x\}$
2. $\{\neg x, \neg x \vee y\}$
3. $\{\neg x, \neg x \vee \neg y\}$

Algorithm: Computing Hitting Sets of MCSs (For a Branch)

$$MCS_1 = \{x\}$$
$$MCS_2 = \{\neg x, \neg x \vee y\}$$
$$MCS_3 = \{\neg x, \neg x \vee \neg y\}$$

- Select a clause to add to the growing set of MUS: *selClause* $= \neg x$, *MUS* $= \neg x$

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Algorithm: Computing Hitting Sets of MCSs (For a Branch)

$$MCS_1 = \{x\}$$
$$MCS_2 = \{\neg x, \neg x \vee y\}$$
$$MCS_3 = \{\neg x, \neg x \vee \neg y\}$$

- Select a clause to add to the growing set of MUS: *selClause* $= \neg x$, *MUS* $= \neg x$
- Select a MCS in which *selClause* appears : *selMCS* $= MCS_2$, *newMCSs* $=$ *MCSs*

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Algorithm: Computing Hitting Sets of MCSs (For a Branch)

$$MCS_1 = \{x\}$$
$$MCS_2 = \{\neg x, \cancel{\neg x \vee y}\}$$
$$MCS_3 = \{\neg x, \neg x \vee \neg y\}$$

- Select a clause to add to the growing set of MUS: *selClause* $= \neg x$, *MUS* $= \neg x$
- Select a MCS in which *selClause* appears : *selMCS* $= MCS_2$, *newMCSs* $=$ *MCSs*
- Remove any other clauses of *selMCS* from each set of MCSs

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Algorithm: Computing Hitting Sets of MCSs (For a Branch)

$MCS_1 = \{x\}$

$\cancel{MCS_2 = \{\neg x\}}$

$\cancel{MCS_3 = \{\neg x, \neg x \vee \neg y\}}$

- Select a clause to add to the growing set of MUS: $selClause = \neg x$, $MUS = \neg x$
- Select a MCS in which *selClause* appears : $selMCS = MCS_2$, $newMCSs = MCSs$
- Remove any other clauses of *selMCS* from each set of MCSs
- Remove MCSs from *newMCSs* in which *selClause* contains.

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

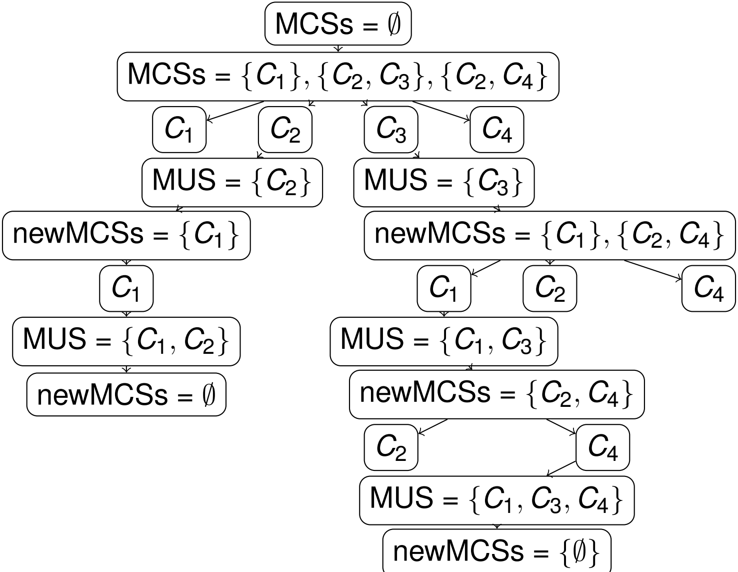
Algorithm: Computing Hitting Sets of MCSs (For a Branch)

$MCS_1 = \{x\}$

- Select a clause to add to the growing set of MUS: $selClause = \neg x$, $MUS = \neg x$
- Select a MCS in which *selClause* appears : $selMCS = MCS_2$, $newMCSs = MCSs$
- Remove any other clauses of *selMCS* from each set of MCSs
- Remove MCSs from *newMCSs* in which *selClause* contains.
- Iterate until $newMCSs = \emptyset$, empty *newMCSs* is found by generating a MUS $\{x, \neg x\}$

$$\varphi = \underbrace{(x)}_{C_1} \wedge \underbrace{(\neg x)}_{C_2} \wedge \underbrace{(\neg x \vee y)}_{C_3} \wedge \underbrace{(\neg x \vee \neg y)}_{C_4}$$

Example: Computing Hitting Sets of MCSs



Thank You!