CHALMERS
UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG

Lecture 4: Feature Model and Code Analysis

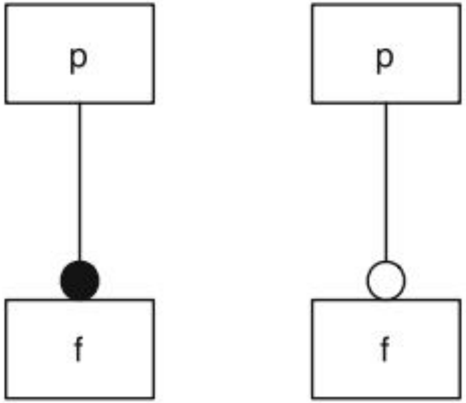
Gregory Gay
TDA 594/DIT 593 - November 11, 2021

Propositional Logic

- **Mandatory:** If parent is selected, the child must be.
  - `mandatory(p, f) ≡ f ⇔ p`
- **Optional:** Child may only be chosen if the parent is.
  - `optional(p, f) ≡ f ⇒ p`

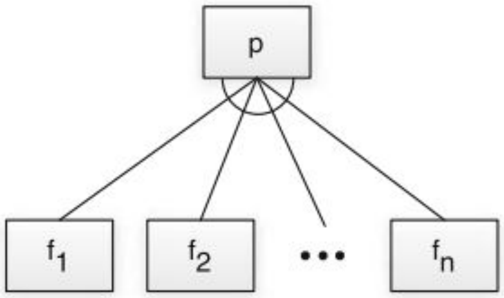
Mandatory Feature

Optional Feature

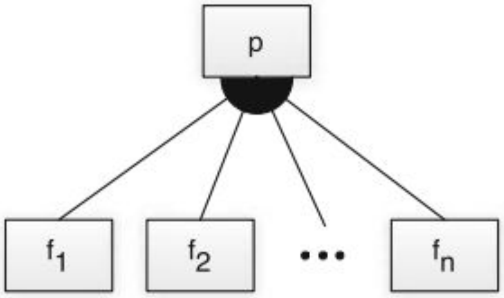


Propositional Logic

- **Alternative:** Choose **exactly** one
  - alternative(p, $\{f_1,...,f_n\}$) ≡ $((f_1 \lor ... \lor f_n) \Leftrightarrow p) \land_{(fi,fj)} \neg(f_i \land f_j)$



- **Or:** Choose **at least** one
  - or(p, $\{f_1,...,f_n\}$) ≡ $((f_1 \lor ... \lor f_n) \Leftrightarrow p)$

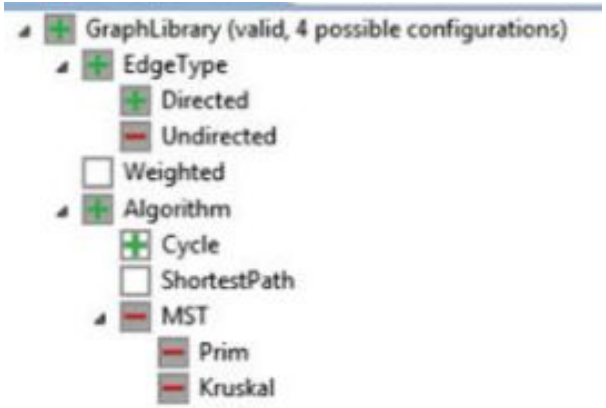


Analyses of Feature Models

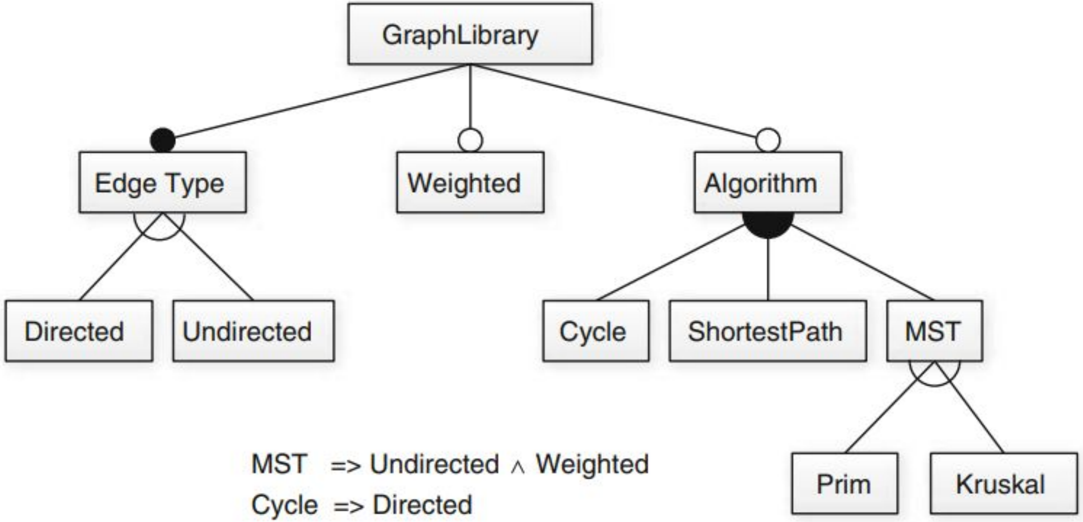
- Is a feature selection valid?
- Is the feature model consistent?
- Do our assumptions hold (testing)?
- Which features are mandatory?
- Which features can never be selected (dead)?
- How many valid selections does model have?
- Are two models equivalent?
- Given partial selection, what must be included?
- What selections give best cost/size/performance?

Valid Feature Selection

- Translate model into a propositional formula φ.
- Assign `true` to each selected feature, `false` to rest.
- Assess whether φ is `true`.
  - If yes, valid selection.

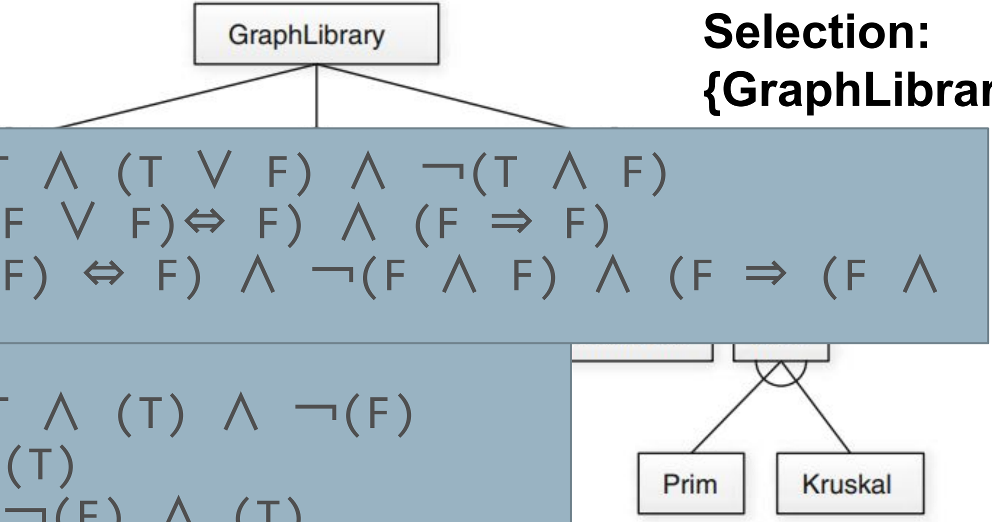


Example - Graph Library



$$\begin{aligned}\phi = &\texttt{GraphLibrary} \wedge \texttt{EdgeType} \wedge (\texttt{Directed} \vee \texttt{Undirected}) \wedge \neg(\texttt{Directed} \wedge \texttt{Undirected}) \\ &\wedge ((\texttt{Cycle} \vee \texttt{ShortestPath} \vee \texttt{MST}) \Leftrightarrow \texttt{Algorithm}) \wedge (\texttt{Cycle} \Rightarrow \texttt{Directed}) \\ &\wedge ((\texttt{Prim} \vee \texttt{Kruskal}) \Leftrightarrow \texttt{MST}) \wedge \neg(\texttt{Prim} \wedge \texttt{Kruskal}) \wedge (\texttt{MST} \Rightarrow (\texttt{Undirected} \wedge \texttt{Weighted}))\end{aligned}$$

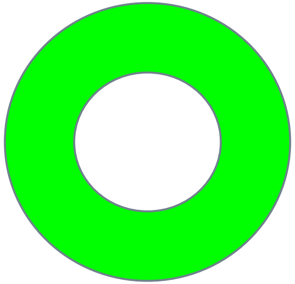
Example - Graph Library



$$\varphi = T \wedge T \wedge (T \vee F) \wedge \neg(T \wedge F) \wedge ((F \vee F \vee F) \Leftrightarrow F) \wedge (F \Rightarrow F) \wedge ((F \vee F) \Leftrightarrow F) \wedge \neg(F \wedge F) \wedge (F \Rightarrow (F \wedge F))$$

$$\varphi = T \wedge T \wedge (T) \wedge \neg(F) \wedge (T) \wedge (T) \wedge (T) \wedge \neg(F) \wedge (T)$$

Selection:
{GraphLibrary, EdgeType, Directed}



$$\phi = \mathtt{GraphLibrary} \wedge \mathtt{EdgeType} \wedge (\mathtt{Directed} \vee \mathtt{Undirected}) \wedge \neg(\mathtt{Directed} \wedge \mathtt{Undirected}) \wedge ((\mathtt{Cycle} \vee \mathtt{ShortestPath} \vee \mathtt{MST}) \Leftrightarrow \mathtt{Algorithm}) \wedge (\mathtt{Cycle} \Rightarrow \mathtt{Directed}) \wedge ((\mathtt{Prim} \vee \mathtt{Kruskal}) \Leftrightarrow \mathtt{MST}) \wedge \neg(\mathtt{Prim} \wedge \mathtt{Kruskal}) \wedge (\mathtt{MST} \Rightarrow (\mathtt{Undirected} \wedge \mathtt{Weighted}))$$

Consistent Feature Models

- A **consistent** model has 1+ valid selections.
  - **Inconsistent** models do not have any valid selection.
- Contradictory constraints are common.
- Find feature selection that results in φ = `true`
  - NP-complete problem, but SAT solvers can often find solutions quickly.

Boolean Satisfiability (SAT)

- Find assignments to Boolean variables $X_1, X_2, \ldots, X_n$ that results in expression φ evaluating to true.
- Defined over expressions written in **conjunctive normal form**.
  - $\varphi = (X_1 \lor \neg X_2) \land (\neg X_1 \lor X_2)$
  - $(X_1 \lor \neg X_2)$ is a **clause**, made of variables, ¬, ∨
  - Clauses are joined with ∧

Conjunctive Normal Form

- Variables: X_1, X_2, X_3, X_4, X_5
- Clauses (using only $\lor$ (or) and $\neg$ (not)):
  - $(\neg X_2 \lor X_5)$, $(X_1 \lor \neg X_3 \lor X_4)$, $(X_4 \lor \neg X_5)$, $(X_1 \lor X_2)$
- Expression φ joins clauses with $\land$ (and)
  - $(\neg X_2 \lor X_5) \land (X_1 \lor \neg X_3 \lor X_4) \land (X_4 \lor \neg X_5) \land (X_1 \lor X_2)$

Boolean Satisfiability

- Find assignment to X_1, X_2, X_3, X_4, X_5 to solve
  - $(\neg X_2 \lor X_5) \land (X_1 \lor \neg X_3 \lor X_4) \land (X_4 \lor \neg X_5) \land (X_1 \lor X_2)$
- One solution: 1, 0, 1, 1, 1
  - $(\neg X_2 \lor X_5) \land (X_1 \lor \neg X_3 \lor X_4) \land (X_4 \lor \neg X_5) \land (X_1 \lor X_2)$
  - $(\neg 0 \lor 1) \land (1 \lor \neg 1 \lor 1) \land (1 \lor \neg 1) \land (1 \lor 0)$
  - $(1) \land (1) \land (1) \land (1)$
  - 1

Transformation Rules

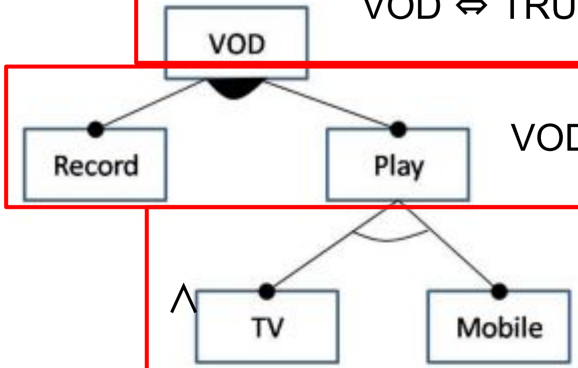
- De Morgan's Laws
  - $\neg(X \lor Y) \equiv \neg X \land \neg Y$
  - $\neg(X \land Y) \equiv \neg X \lor \neg Y$
- Distributivity
  - $X \lor (Y \land Z) \equiv (X \lor Y) \land (X \lor Z)$
  - $X \land (Y \lor Z) \equiv (X \land Y) \lor (X \land Z)$
- Double Negation
  - $\neg\neg X \equiv X$

Transformation Rules

- $X \Leftrightarrow Y$
  - X is equivalent to Y
- $\equiv (X \Rightarrow Y) \land (Y \Rightarrow X)$
  - $(X \Rightarrow Y) \equiv (\neg X \lor Y)$
  - If X is true, Y is also true.
  - If X is false, Y can be either true or false.
- $\equiv (\neg X \lor Y) \land (\neg Y \lor X)$

Transformation into CNF

VOD ⇔ TRUE ≡ VOD



VOD ⇔ (Record ∨ Play)

((Mobile ∨ TV) ⇔ Play)

∧ ¬(Movile ∧ TV) ≡

(Mobile ⇔ Play ∧ ¬TV) ∧ (TV ⇔ Play ∧ ¬Mobile)

mandatory(p, f) ≡ f ⇔ p

optional(p, f) ≡ f ⇒ p

alternative(p, $\{f_1,\ldots,f_n\}$) ≡ $((f_1 \lor \ldots \lor f_n) \Leftrightarrow p) \land \forall(f_i,f_j)\, \neg(f_i \land f_j)$

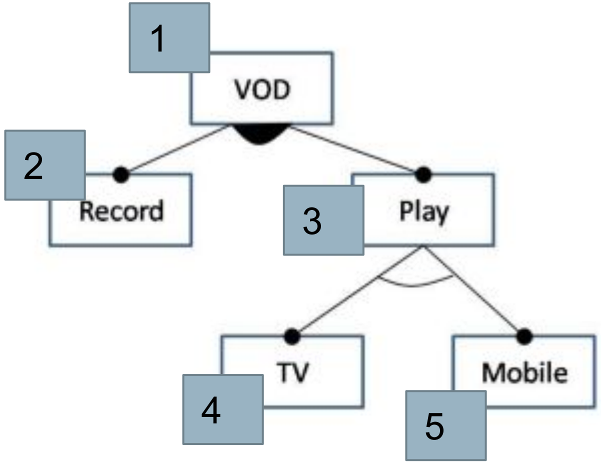
or(p, $\{f_1,\ldots,f_n\}$) ≡ $((f_1 \lor \ldots \lor f_n) \Leftrightarrow p)$

VOD ∧ (VOD ⇔ (Record ∨ Play)) ∧ (Mobile ⇔ Play ∧ ¬TV) ∧ (TV ⇔ Play ∧ ¬Mobile)

Transformation into CNF

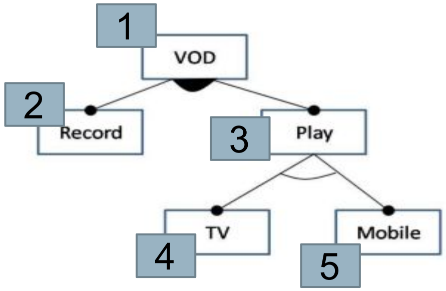
- VOD ∧ (VOD ⇔ (Record ∨ Play)) ∧ (Mobile ⇔ (Play ∧ ¬TV)) ∧ (TV ⇔ (Play ∧ ¬Mobile))
  - (VOD ⇔ (Record ∨ Play))
    - ≡ (VOD ⇒ (Record ∨ Play)) ∧ ((Record ∨ Play) ⇒ VOD)
    - ≡ (¬VOD ∨ (Record ∨ Play)) ∧ (¬(Record ∨ Play) ∨ VOD)
    - ≡ (¬VOD ∨ (Record ∨ Play)) ∧ (¬Record ∨ VOD) ∧ (¬Play ∨ VOD)
  - (Mobile ⇔ (Play ∧ ¬TV))
    - ≡ (Mobile ∨ TV ∨ ¬Play) ∧ (¬Mobile ∨ Play) ∧ (¬Mobile ∨ ¬TV)
  - (TV ⇔ (Play ∧ ¬Mobile))
    - ≡ (TV ∨ Mobile ∨ ¬Play) ∧ (¬TV ∨ Play) ∧ (¬TV ∨ ¬Mobile)

DIMACS Format



- Map feature names to integer IDs.
  - VOD = 1
  - Record = 2
  - Play = 3
  - TV = 4
  - Mobile = 5

DIMACS Format



VOD ∧

(¬VOD ∨ (Record ∨ Play)) ∧ (¬Record ∨ VOD) ∧ (¬Play ∨ VOD) ∧

(Mobile ∨ TV ∨ ¬Play) ∧ (¬Mobile ∨ Play) ∧ (¬Mobile ∨ ¬TV) ∧

(TV ∨ Mobile ∨ ¬Play) ∧ (¬TV ∨ Play) ∧ (¬TV ∨ ¬Mobile)

1 ∧

(¬1 ∨ (2 ∨ 3)) ∧ (¬2 ∨ 1) ∧ (¬3∨ 1) ∧

(5 ∨ 4 ∨ ¬3) ∧ (¬5 ∨ 3) ∧ (¬5 ∨ ¬4) ∧

(4 ∨ 5 ∨ ¬3) ∧ (¬4 ∨ 3) ∧ (¬4∨ ¬5)

DIMACS Format

1 ∧

(¬1 ∨ (2 ∨ 3)) ∧ (¬2 ∨ 1) ∧ (¬3 ∨ 1) ∧

(5 ∨ 4 ∨ ¬3) ∧ (¬5 ∨ 3) ∧ (¬5 ∨ ¬4) ∧

(4 ∨ 5 ∨ ¬3) ∧ (¬4 ∨ 3) ∧ (¬4 ∨ ¬5)

- Each clause is stored in a row, with ∧(AND) omitted.
- Negation (¬) translated into negative (-)

1

-1 ∨ 2 ∨ 3

-2 ∨ 1

-3 ∨ 1

5 ∨ 4 ∨ -3

-5 ∨ 3

-5 ∨ -4

4 ∨ 5 ∨ -3

-4 ∨ 3

-4 ∨ -5

```
1
-1 V 2 V 3
-2 V 1
-3 V 1
5 V 4 V -3
-5 V 3
-5 V -4
4 V 5 V -3
-4 V 3
-4 V -5
```

- Remove disjunction signs (V)
- Add DIMACs header
  - Comments
  - Indicates CNF format
  - Number of variables
  - Number of CNF clauses

```
c comments
p cnf 5 10
1
-1 2 3
-2 1
-3 1
5 4 -3
-5 3
-5 -4
4 5 -3
-4 3
-4-5
```

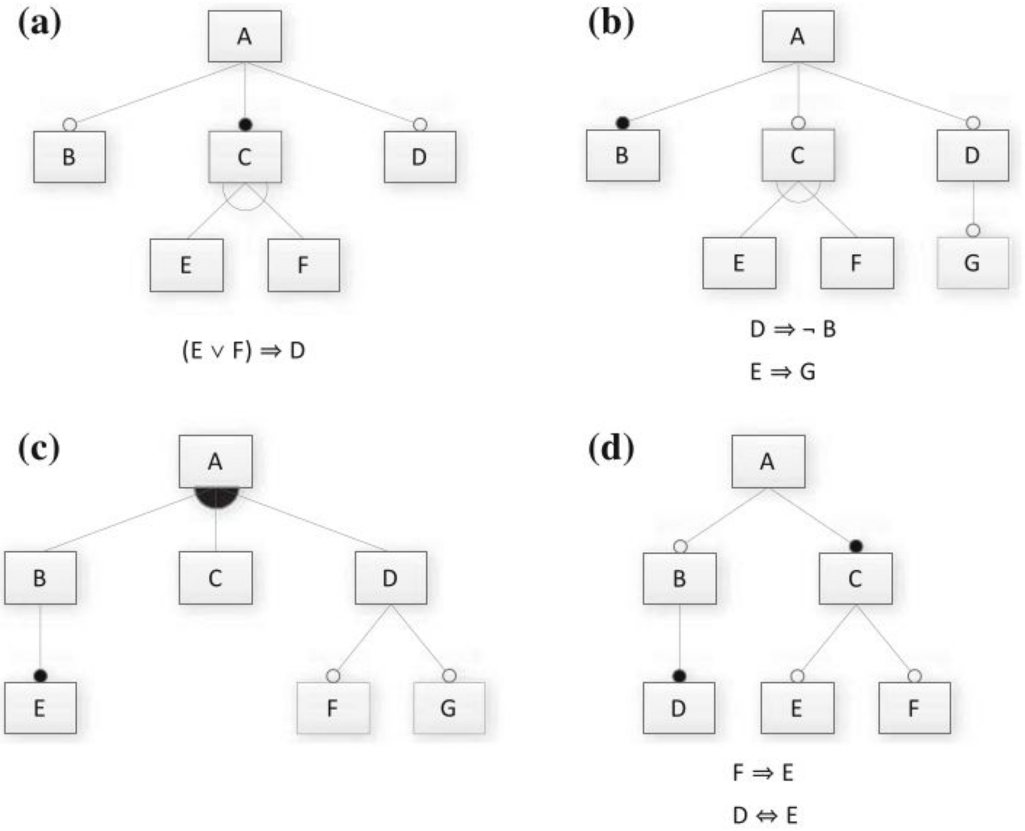
Using a SAT Solver

- Identify assignment that results in true outcome.
  - VOD ∧ (¬VOD ∨ (Record ∨ Play)) ∧ (¬Record ∨ VOD) ∧ (¬Play ∨ VOD) ∧ (Mobile ∨ TV ∨ ¬Play) ∧ (¬Mobile ∨ Play) ∧ (¬Mobile ∨ ¬TV) ∧ (TV ∨ Mobile ∨ ¬Play) ∧ (¬TV ∨ Play) ∧ (¬TV ∨ ¬Mobile)
  - A satisfying assignment: (1, 1, 1, 1, 0)
- Returns satisfying assignment.
  - May return all satisfying assignments found.
  - If not satisfiable, may offer information on why.

https://bit.ly/3BVwMZc

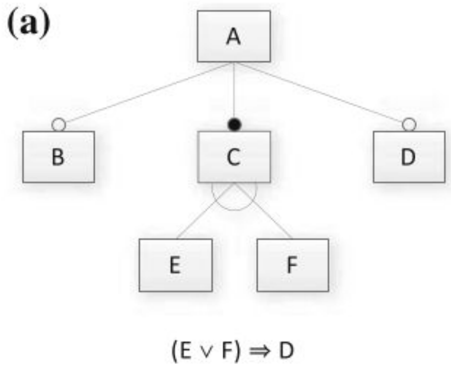
Activity

- Start with A/B.
  - Do C/D if time.
- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?



Solution (A)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?

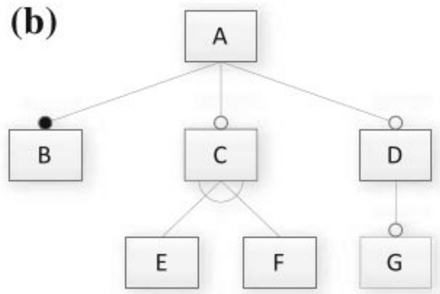


A ∧ (B ⇒ A) ∧ (C ⇔ A) ∧ (D ⇒ A) ∧
((C ⇔ (E ∨ F)) ∧ ¬(E ∧ F)) ∧ ((E ∨ F) ⇒ D))

- Valid: A, B, C, D, F ; A, C, D, E
- Invalid: A, B, C, D, E, F ; A, B, C, E
- Is it consistent: Yes

Solution (B)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?



A ∧ (B ⇔ A) ∧ (C ⇒ A) ∧ (D ⇒ A) ∧
((C ⇔ (E ∨ F)) ∧ ¬(E ∧ F)) ∧ (G ⇒ D) ∧ (D ⇒ ¬B)
∧
(E ⇒ G)

- Valid: A, B ; A, B, C, F
- Invalid: A, B, D, G ; A, B, C, E
- It is consistent: Yes, but D, E, and G are dead features (because B is mandatory).

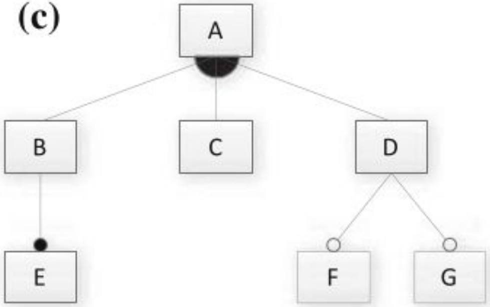
Solution (C)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?

A ∧ ((B ∨ C ∨ D) ⇔ A) ∧ (E ⇔ B) ∧ (F ⇒ D) ∧ (G ⇒ D)

- Valid: A, C ; A, B, C, D, E, F, G
- Invalid: A, B, C; A, C, E
- It is consistent: Yes (just remember that B and E need to come as a pair)

(c)



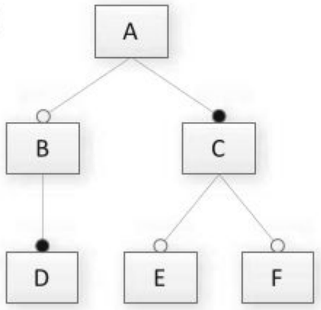
Solution (D)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?

$A \land (B \Rightarrow A) \land (C \Leftrightarrow A) \land (D \Leftrightarrow B) \land (E \Rightarrow C) \land (F \Rightarrow C) \land$
$(F \Rightarrow E) \land (D \Leftrightarrow E)$

- Valid: A, C ; A, B, C, D, E
- Invalid: A, B, C, D ; A, C, F
- It is consistent: Yes, but remember that if you have F, you need E, D, and B as well.

(d)



$F \Rightarrow E$

$D \Leftrightarrow E$

SAT Solver Process

- Express in conjunctive normal form:
  - $\varphi = (\neg x2 \lor x5) \land (x1 \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (x1 \lor x2)$
- Choose assignment based on how it affects each clause it appears in.
  - What happens if we assign x2 = true?
  - If any clauses now false, don’t apply that value.
  - Continue until CNF expression is satisfied.

Branch & Bound Algorithm

- Set variable to true or false.
- Apply that value.
- Does value satisfy the clauses that it appears in?
  - If so, assign a value to the next variable.
  - If not, backtrack (bound) and apply the other value.
- Prunes branches of the boolean decision tree as values are applied.

Branch & Bound Algorithm

$\varphi = (\neg x2 \lor x5) \land (x1 \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (x1 \lor x2)$

1. **Set x1 to false.**
   $\varphi = (\neg x2 \lor x5) \land (\mathbf{0} \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (\mathbf{0} \lor x2)$
2. **Set x2 to false.**
   $\varphi = (\mathbf{1} \lor x5) \land (\mathbf{0} \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (\mathbf{0} \lor \mathbf{0})$
3. **Backtrack and set x2 to true.**
   $\varphi = (\mathbf{0} \lor x5) \land (\mathbf{0} \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (\mathbf{0} \lor \mathbf{1})$

DPLL Algorithm

- Set a variable to true/false.
  - Apply that value to the expression.
  - Remove all satisfied clauses.
  - If assignment does not satisfy a clause, then remove that variable from that clause.
  - If this leaves any **unit clauses** (single variable clauses), assign a value that removes those next.
- Repeat until a solution is found.

DPLL Algorithm

$\varphi = (\neg x2 \lor x5) \land (x1 \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (x1 \lor x2)$

1. **Set x2 to false.**
   $\varphi = (\neg \mathbf{0} \lor x5) \land (x1 \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (x1 \lor \mathbf{0})$
   $\varphi = (x1 \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (x1)$
2. **Set x1 to true.**
   $\varphi = (\mathbf{1} \lor \neg x3 \lor x4) \land (x4 \lor \neg x5) \land (\mathbf{1})$
   $\varphi = (x4 \lor \neg x5)$
3. **Set x4 to false, then x5 to false.**
   $\varphi = (\mathbf{0} \lor \neg x5)$
   $\varphi = (\neg \mathbf{0})$

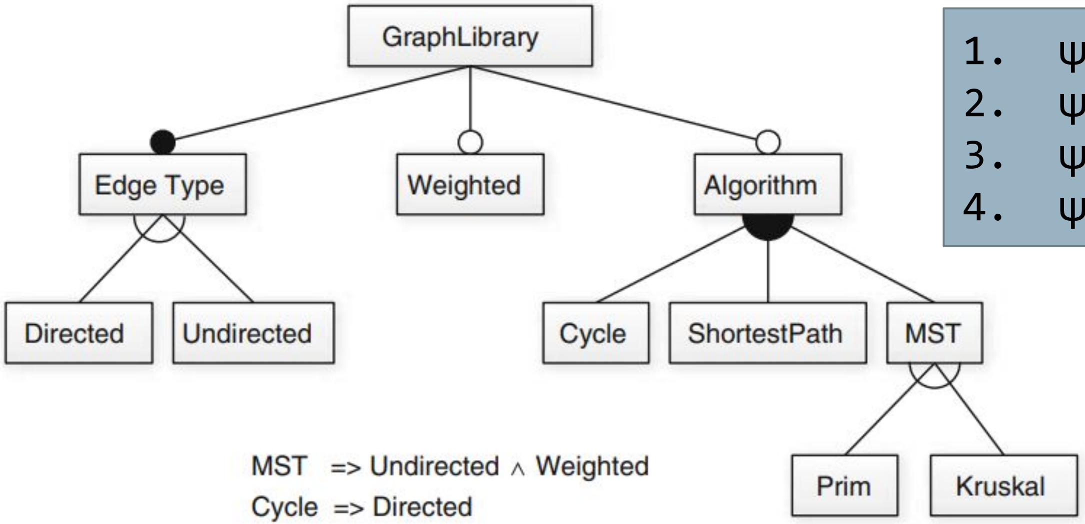
Let’s take a break!

Testing Facts About the Model

Testing Facts about Models

- A fact that *should be true* encoded as formula ψ.
- Check whether $\varphi \land \neg\psi$ is satisfiable.
  - Is there a valid feature selection for φ that does not satisfy constraint ψ?
  - If yes, there is a problem with the model.

Example - Graph Library



1. ψ = Kruskal ⇒ Weighted
2. ψ = Prim ⇒ Weighted
3. ψ = ¬(Prim ∧ Kruskal)
4. ψ = Weighted ⇒ MST

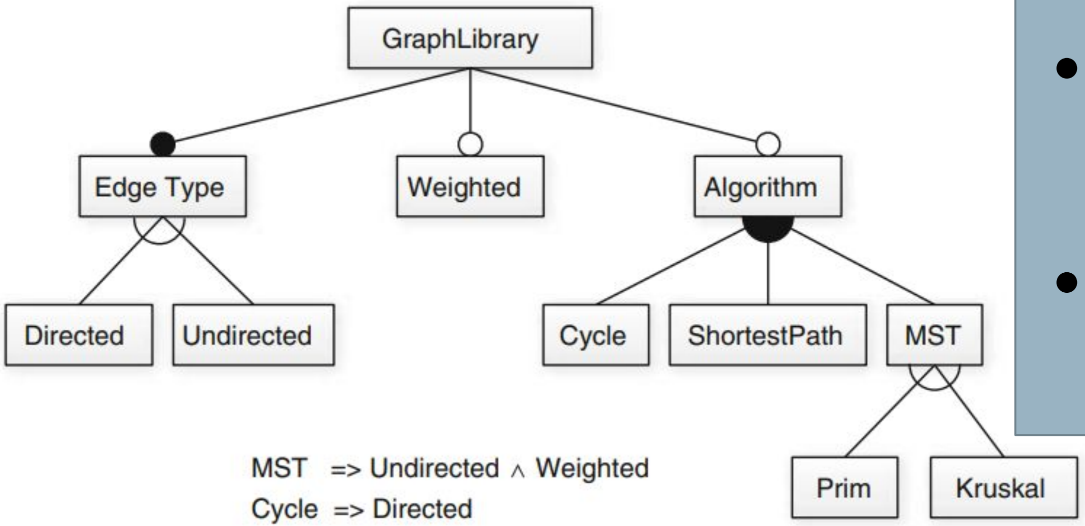
For each, assess whether ($\varphi \wedge \neg\psi$) is satisfiable.

$$\begin{aligned}\phi = &\mathrm{GraphLibrary} \wedge \mathrm{EdgeType} \wedge (\mathrm{Directed} \vee \mathrm{Undirected}) \wedge \neg(\mathrm{Directed} \wedge \mathrm{Undirected})\\ &\wedge ((\mathrm{Cycle} \vee \mathrm{ShortestPath} \vee \mathrm{MST}) \Leftrightarrow \mathrm{Algorithm}) \wedge (\mathrm{Cycle} \Rightarrow \mathrm{Directed})\\ &\wedge ((\mathrm{Prim} \vee \mathrm{Kruskal}) \Leftrightarrow \mathrm{MST}) \wedge \neg(\mathrm{Prim} \wedge \mathrm{Kruskal}) \wedge (\mathrm{MST} \Rightarrow (\mathrm{Undirected} \wedge \mathrm{Weighted}))\end{aligned}$$

Dead and Mandatory Features

- A **dead** feature is never used.
- A **mandatory** feature is always used.
- Given model φ and feature F:
  - 1+ valid selection **with F if (φ ∧ F)** is satisfiable.
  - 1+ valid selection **without F if (φ ∧ ¬F)** is satisfiable.
  - Feature is dead if no selection with it (**¬(φ ∧ F)**)
  - Feature is mandatory if no selection without it (**¬(φ ∧ ¬F)**)

Example - Graph Library



- No dead features.
  - If Undirected made mandatory, Directed and Cycle would be dead.
- GraphLibrary and EdgeType are mandatory.

$$\phi = \mathtt{GraphLibrary} \wedge \mathtt{EdgeType} \wedge (\mathtt{Directed} \vee \mathtt{Undirected}) \wedge \neg(\mathtt{Directed} \wedge \mathtt{Undirected})$$
$$\wedge ((\mathtt{Cycle} \vee \mathtt{ShortestPath} \vee \mathtt{MST}) \Leftrightarrow \mathtt{Algorithm}) \wedge (\mathtt{Cycle} \Rightarrow \mathtt{Directed})$$
$$\wedge ((\mathtt{Prim} \vee \mathtt{Kruskal}) \Leftrightarrow \mathtt{MST}) \wedge \neg(\mathtt{Prim} \wedge \mathtt{Kruskal}) \wedge (\mathtt{MST} \Rightarrow (\mathtt{Undirected} \wedge \mathtt{Weighted}))$$

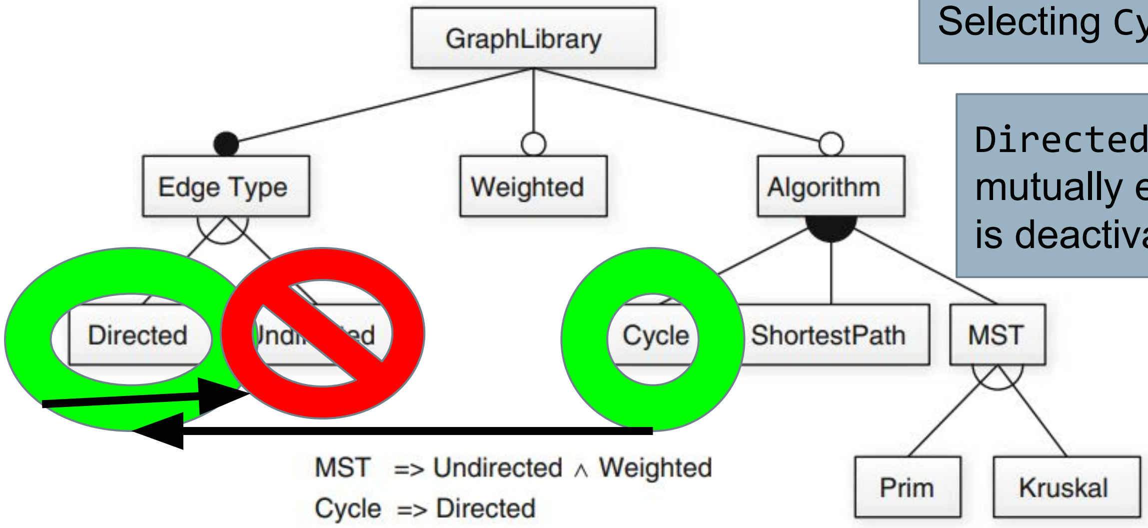
Constraint Propagation

- **Constraint Propagation** - hiding unavailable features after we make **partial selections**.
- Feature selection often iterative:
  - Feature selected, deselected, or no decision made.
- Partial feature selection:
  - Set of selected features ($S \subseteq F$)
  - Set of deselected features ($D \subseteq F$, with $S \cap D = \varnothing$)

Constraint Propagation

- Partial feature selection
  - pfs(S,D) = ∀(s∈S) s ∧ ∀(d∈D) ¬d
- Partial selection is valid if **(φ ∧ pfs(S,D))** satisfiable
- F **deactivated** if **(φ ∧ pfs(S,D) ∧ F)** **not** satisfiable.
- F **activated** if **(φ ∧ pfs(S,D) ∧ ¬F)** **not** satisfiable.

Example - Graph Library

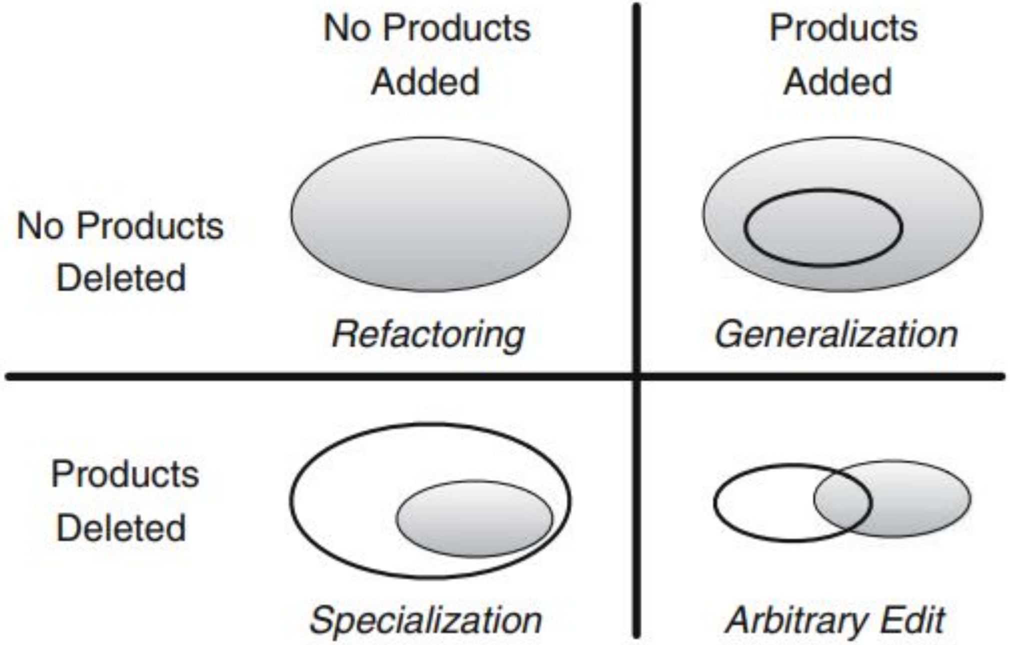


Selecting `Cycle` activated `Directed`.

`Directed` and `Undirected` are mutually exclusive, so `Undirected` is deactivated.

$$\phi = \texttt{GraphLibrary} \wedge \texttt{EdgeType} \wedge (\texttt{Directed} \vee \texttt{Undirected}) \wedge \neg(\texttt{Directed} \wedge \texttt{Undirected})$$
$$\wedge ((\texttt{Cycle} \vee \texttt{ShortestPath} \vee \texttt{MST}) \Leftrightarrow \texttt{Algorithm}) \wedge (\texttt{Cycle} \Rightarrow \texttt{Directed})$$
$$\wedge ((\texttt{Prim} \vee \texttt{Kruskal}) \Leftrightarrow \texttt{MST}) \wedge \neg(\texttt{Prim} \wedge \texttt{Kruskal}) \wedge (\texttt{MST} \Rightarrow (\texttt{Undirected} \wedge \texttt{Weighted}))$$

Comparing Feature Models



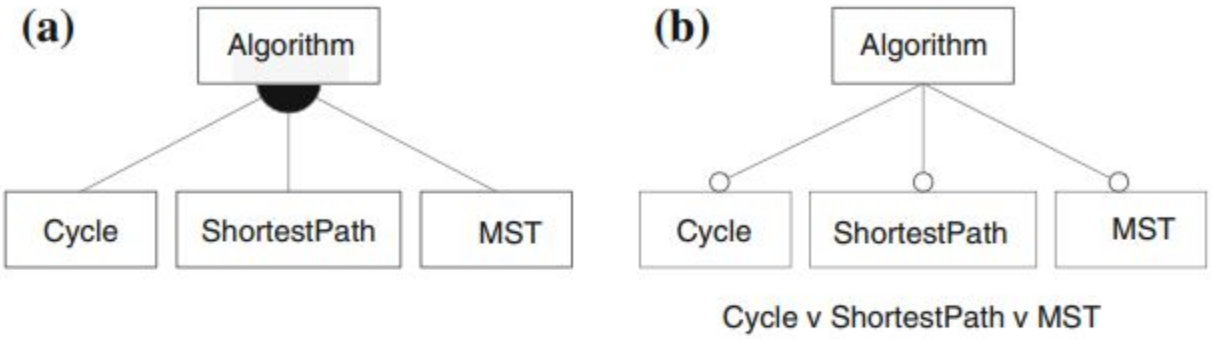
Comparing Feature Models

- Models are equivalent if formulae are equivalent.
  - $\neg(\varphi_1 \Leftrightarrow \varphi_2)$ is **not satisfiable**.
- φ_1 is a specialization of φ_2 if $(\varphi_2 \Rightarrow \varphi_1)$
  - and φ_2 is a generalization of φ_1
- SAT solver can compare two models.

Example - Graph Library

Use SAT Solver to prove

$\varphi_{right} \Leftrightarrow \varphi_{left}$



$$\phi_{\texttt{left}} = \text{Algorithm} \wedge ((\text{Cycle} \vee \text{ShortestPath} \vee \text{MST}) \Leftrightarrow \text{Algorithm})$$

$$\phi_{\texttt{right}} = \text{Algorithm} \wedge (\text{Cycle} \Rightarrow \text{Algorithm}) \wedge (\text{ShortestPath} \Rightarrow \text{Algorithm}) \wedge (\text{MST} \Rightarrow \text{Algorithm}) \wedge (\text{Cycle} \vee \text{ShortestPath} \vee \text{MST})$$

Feature-to-Code Mappings

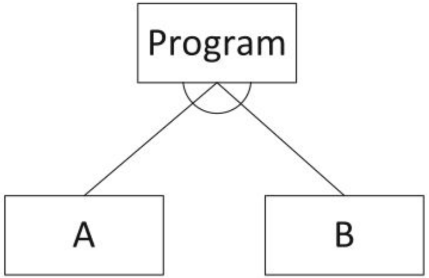
Feature-To-Code Mappings

- Feature models describe the problem space.
- Models are implemented in source code.
- Similar analyses can examine mapping of feature models to code.
  - Which code assets are never used?
  - Which code assets are always used?
  - Which features have no influence on product portfolio?

Dead Code

- Features that can never be incorporated.
- Feature B, in the code, required Feature A to also be selected.
- Model states that A and B are mutually exclusive.

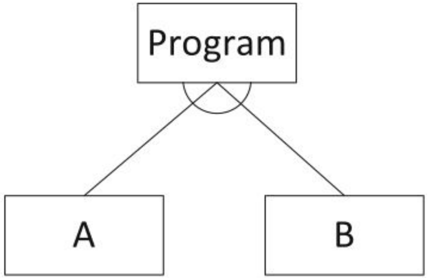
```
line 1
#ifdef A
line 3
#ifdef B
line 5
#endif
#else
line 8
#endif
```



Presence Conditions

- Describes the set of products containing a code fragment.
- **pc(c) = (conditions for c to be included in a product)**
  - pc(line 3) = A
  - pc(line 5) = A ∧ B
  - pc(line 8) = ¬ A

```
1 line 1
2 #ifdef A
3 line 3
4 #ifdef B
5 line 5
6 #endif
7 #else
8 line 8
9 #endif
```

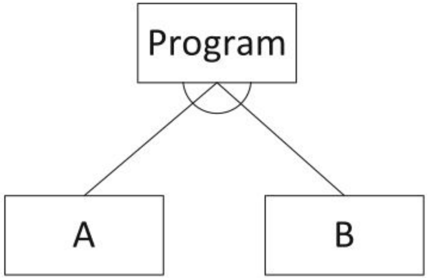


- pc(lines 3-5) = A ∧ B
- pc(lines 3-8) = A ∧ B ∧ ¬A
  - (cannot be included in any product)

Dead Code

- Fragment is dead if never included in any product.
  - φ represents all valid products.
  - Fragment C is dead iff (**φ ∧ pc(C)**) is not satisfiable.

	C	pc()
1	line 1	True
2	#ifdef A	
3	line 3	A
4	#ifdef B	
5	line 5	A ∧ B
6	#endif	
7	#else	
8	line 8	¬A
9	#endif	



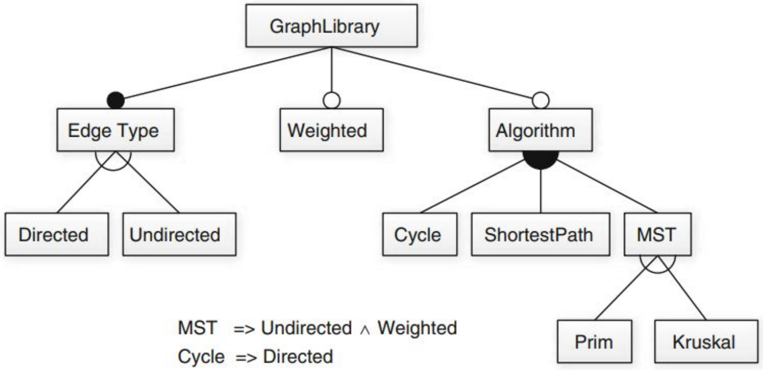
φ = Program ∧ (A ∨ B) ∧ ¬(A ∧ B)

(φ ∧ pc(line 5)) is not satisfiable:

Program ∧ (A ∨ B) ∧ ¬(A ∧ B) ∧ (A ∧ B)

Mandatory Code

- Fragment is mandatory if always included in a product.
  - φ represents all valid products.
  - Fragment C is mandatory iff (**φ ∧ ¬pc(C)**) is **not** satisfiable.



$$\phi = \texttt{GraphLibrary} \land \texttt{EdgeType} \land (\texttt{Directed} \lor \texttt{Undirected}) \land \neg(\texttt{Directed} \land \texttt{Undirected})$$
$$\land ((\texttt{Cycle} \lor \texttt{ShortestPath} \lor \texttt{MST}) \Leftrightarrow \texttt{Algorithm}) \land (\texttt{Cycle} \Rightarrow \texttt{Directed})$$
$$\land ((\texttt{Prim} \lor \texttt{Kruskal}) \Leftrightarrow \texttt{MST}) \land \neg(\texttt{Prim} \land \texttt{Kruskal}) \land (\texttt{MST} \Rightarrow (\texttt{Undirected} \land \texttt{Weighted}))$$

If code implemented correctly, the fragment for EdgeType will be mandatory.

We Have Learned

- Feature Models can be expressed using propositional logic formulae (φ).
  - Based on model and cross-tree constaints.
- Valid feature selections result in (φ = true).
- SAT Solvers can identify valid configurations.
  - If none can be found, the model is inconsistent.
  - Enables many different model analyses.

We Have Learned

- Feature-Model Analysis
  - Check properties of model are true.
  - Dead and mandatory features
  - Effects of partial selections
  - Comparisons between two models
- Mapping of models and code
  - Dead and mandatory code

Next Time

- Variability Implementation
- Assignment 1
  - Due November 14
  - Reach out to supervisors (and me) with questions
- Assignment 2
  - Due November 21
  - Feature modelling and analysis for mobile robots

UNIVERSITY OF GOTHENBURG

CHALMERS

UNIVERSITY OF TECHNOLOGY