Lecture 3: Feature Modeling

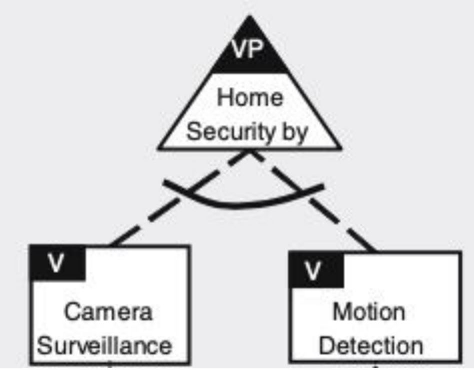
Gregory Gay
TDA 594/DIT 593 - November 8, 2022

Today’s Goals

- Feature Modeling
  - Feature Diagrams
  - Propositional Logic
- Analysis of Feature Models

Reasoning about Variability

- **Variation Point**
  - Where one product can differ from another.
  - Ex: Which features are supported by this security alarm?



- **Feature**
  - Options that can be chosen at each variation point.
  - Ex: Motion detection, camera

Features and Products

- Any end-user-visible characteristic or behavior of a system is a **feature**.
  - (often, functionality a user can directly interact with)
- A concrete **product** is a valid **feature selection**.
  - Fulfills all **variability and feature dependencies.**

Feature Modeling

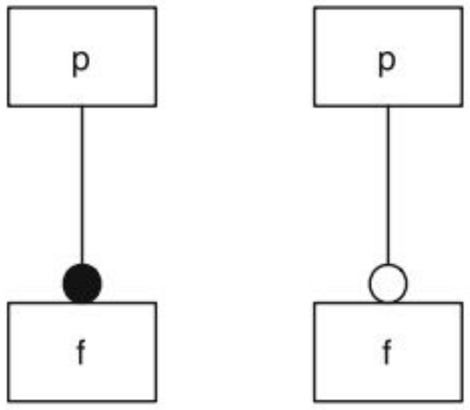
- A specification of variation points and features in a hierarchical form.
  - Represented visually using **feature diagrams**.
  - Also represented as **propositional logic** for analysis.
- Enables understanding of dependencies and what valid products can be built using a platform.

Feature Diagrams

Feature Diagrams

Mandatory Feature

Optional Feature

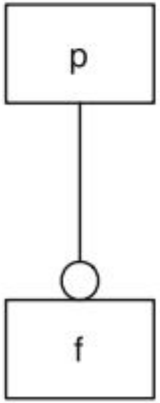


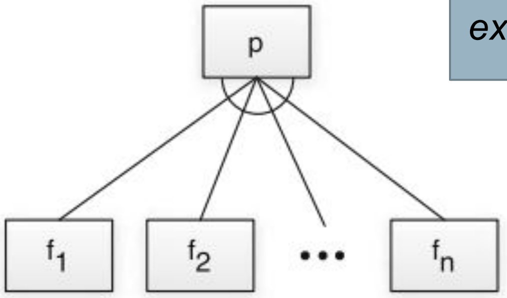
- Tree where nodes represent features.
- Shows parent-child relationship.
  - F can only be selected when P is selected.
  - Parent tends to be more general, child is more specific.
    - Parent - Sensor, Child - RADAR

Feature Diagrams

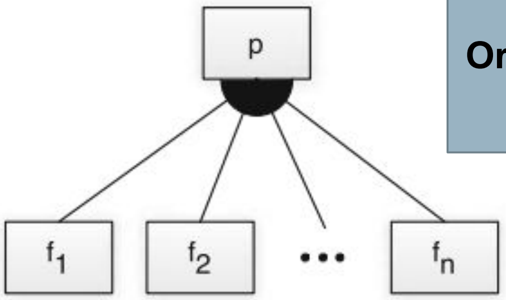
Mandatory Feature

Optional Feature





Alternative (mutually exclusive choice): Choose *exactly* one



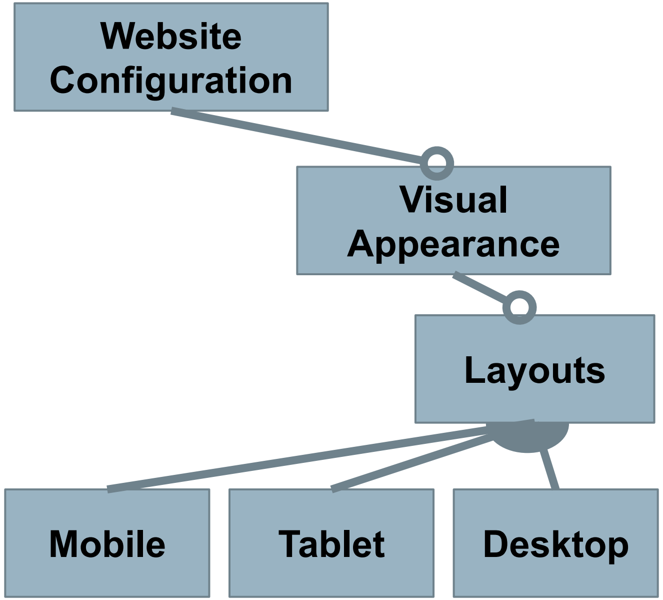
Or: Choose **at least** one

Cross-Tree Constraints

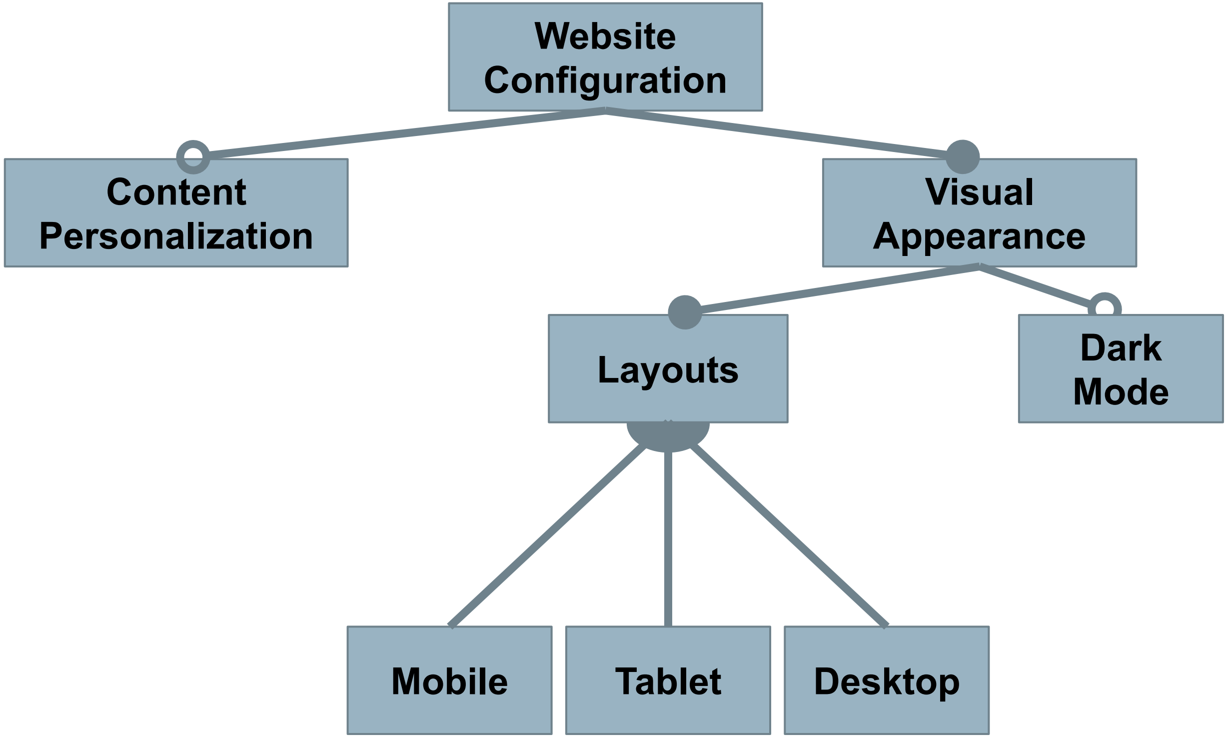
- **Cross-tree Constraints** are predicates imposing constraints between features.
  - `DataDictionary` $\Rightarrow$ `String`
    - (Storing a data dictionary **requires** support for strings)
  - `MinimumSpanningTree` $\Rightarrow$ `Undirected` $\wedge$ `Weighted`
    - (Computing a Minimum Spanning Tree **requires** support for undirected **and** weighted edges)
  - Constraints over Boolean variables and subexpressions.
    - (i.e., `(NumProcesses >= 5)`)

Example - Website Configuration

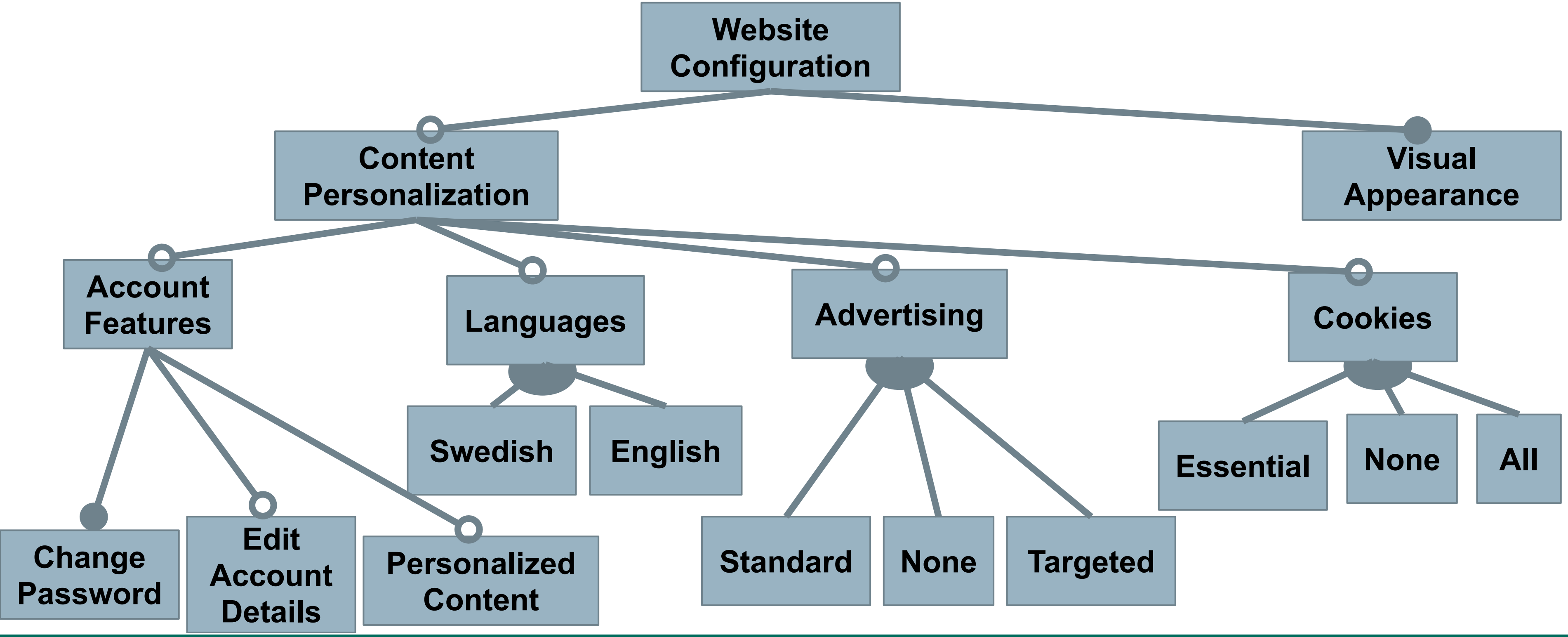
- SPL that provides website functionality.
- One feature - adjusts layout based on the device.
- What other aspect of the site could be features?
  - Consider visual appearance and personalized content.



Example - Website Configuration



Example - Website Configuration



Activity - Smart TV OS

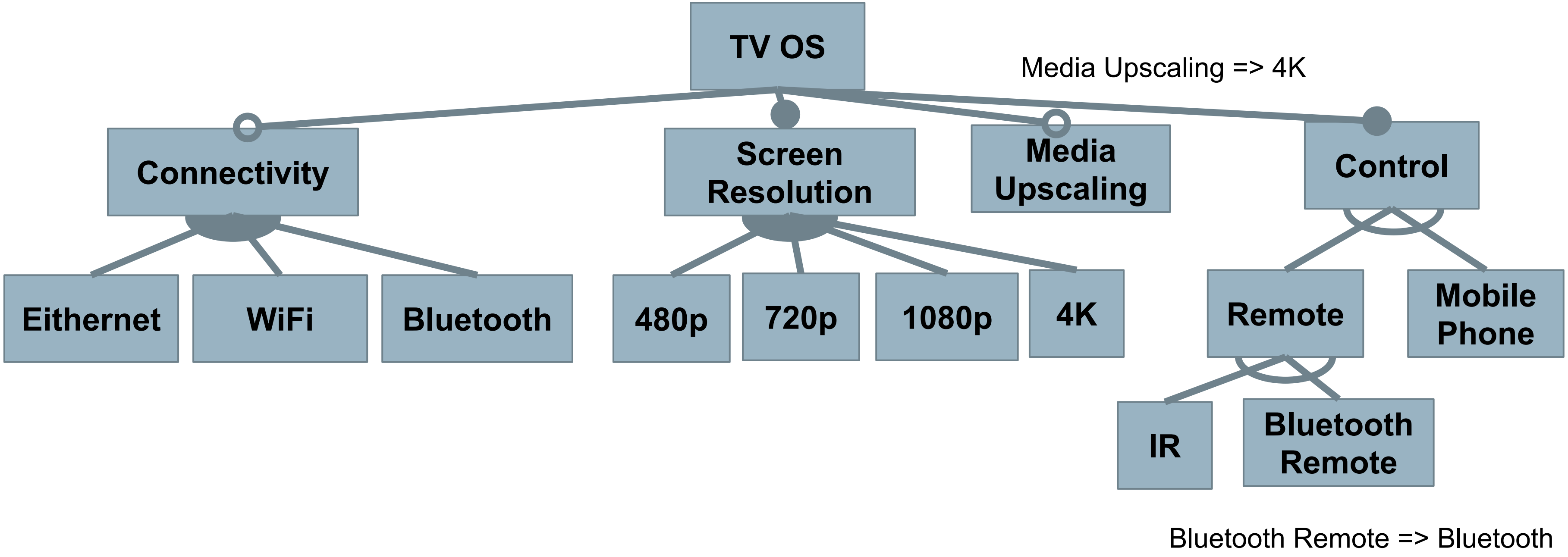
- Your company is developing a product line of smart televisions, with different feature configurations.
  - Identify the features of this product line.
  - Model the domain with a feature diagram.
- Consider existing products on the market (e.g., Samsung TVs). Maybe check an electronics website (elgiganten, etc.).

Activity - Smart TV OS

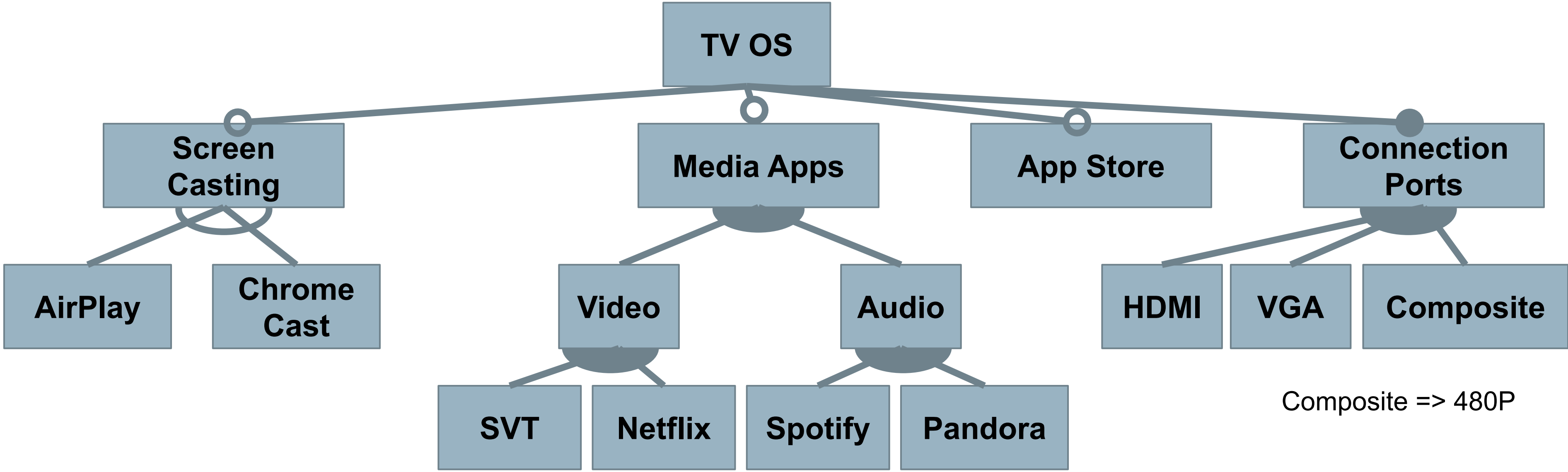
- Which features will many (or few) customers want?
- Which features might distinguish your product from others on the market?
- Don’t try to capture all features, but an interesting subset (aim for 15-25).
- Capture dependencies between features using visual structures and cross-tree constraints.

Activity - Smart TV OS

Possible Solution



Possible Solution



Composite => 480P

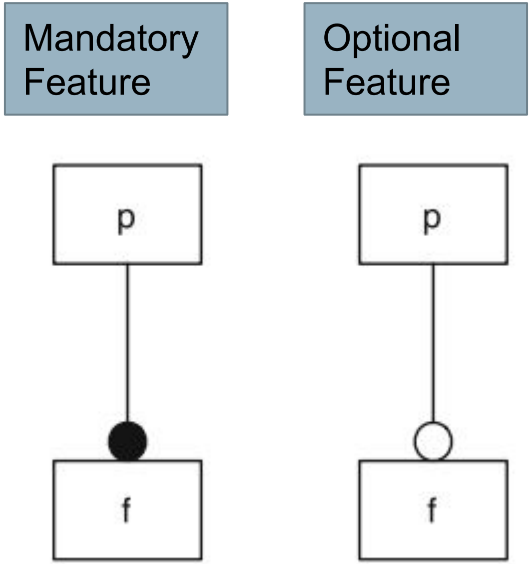
Screen Casting => Connectivity
Media Apps => Connectivity
App Store => Connectivity

Let’s take a break!

Propositional Logic and Feature Model Analysis

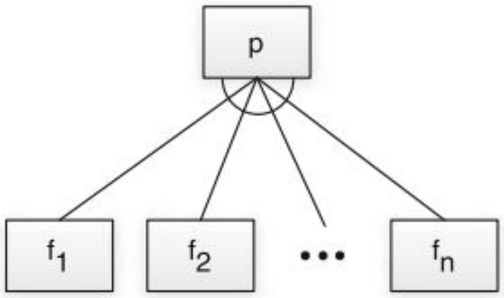
Propositional Logic

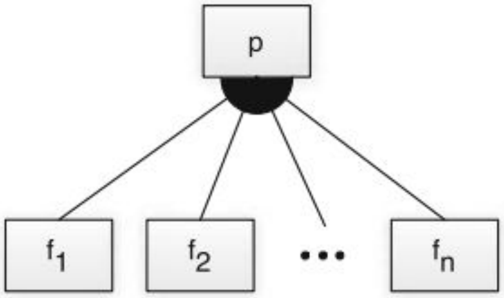
- **Mandatory:** If parent is selected, the child must be.
  - $\mathrm{mandatory}(p, f) \equiv f \Leftrightarrow p$
- **Optional:** Child may only be chosen if the parent is.
  - $\mathrm{optional}(p, f) \equiv f \Rightarrow p$



Propositional Logic

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

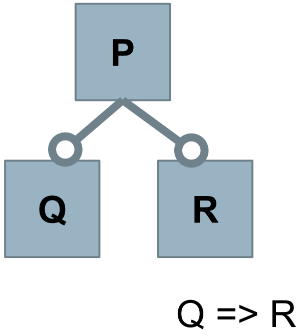




Propositional Logic

- Cross-tree constraints already expressed in logic.
- Form a single formula capturing how products are configured by joining each node relationship and cross-tree constraint using AND ($\land$)

$$
\begin{aligned}
\varphi = \; & P \\
& \land (Q \Rightarrow P) \\
& \land (R \Rightarrow P) \\
& \land (Q \Rightarrow R)
\end{aligned}
$$

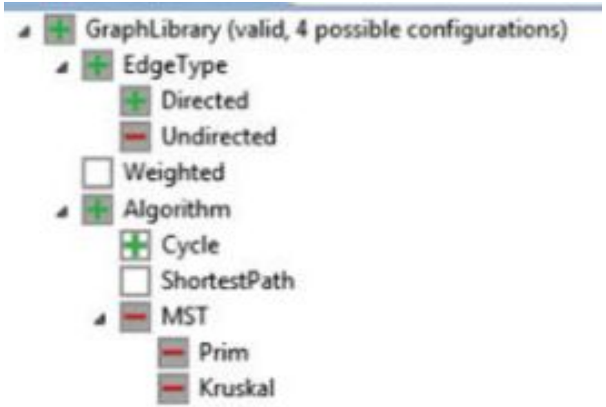


Analyses of Feature Models

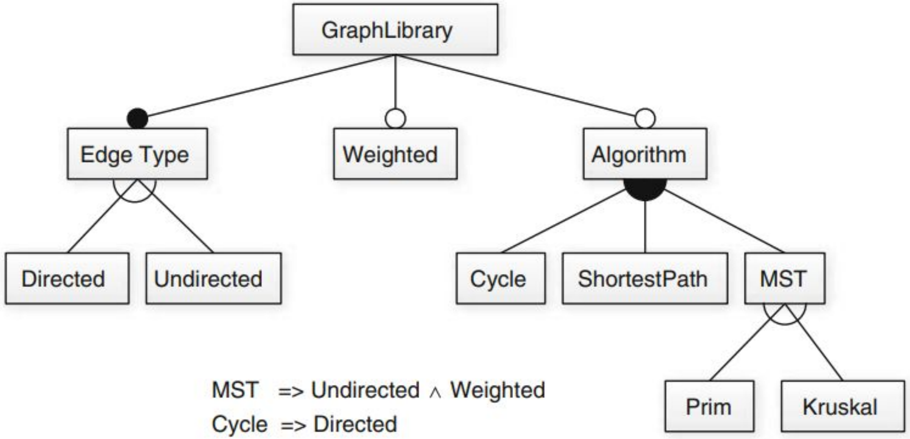
- Is a feature selection valid?
- Is the feature model consistent?
- Do our assumptions hold (testing)?
- Which features are mandatory?
- Which features can never be selected (dead)?
- How many valid selections does model have?
- Are two models equivalent?
- Given partial selection, what must be included?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

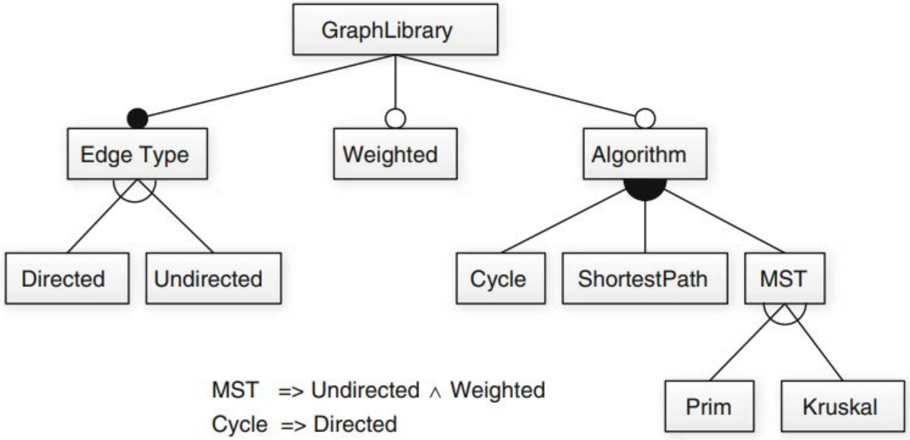


Example - Graph Library



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

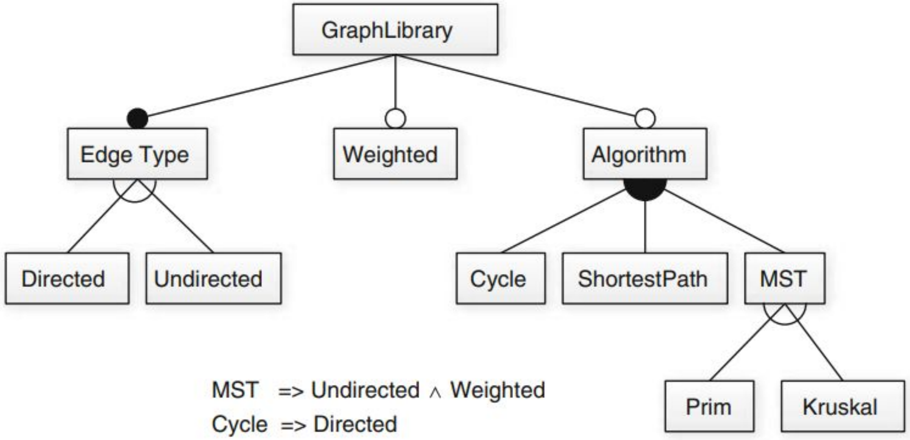
Example - Graph Library

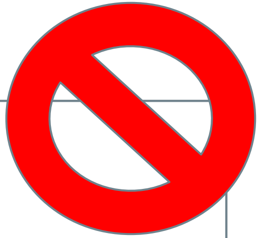


Selection:
{GraphLibrary, EdgeType, Directed}

$$
\begin{aligned}
\varphi &= T \\
&\land (T \Leftrightarrow T) \\
&\land (F \Rightarrow T) \\
&\land (F \Rightarrow T) \\
&\land (((T) \Leftrightarrow T) \land T) \\
&\land ((F) \Leftrightarrow F) \\
&\land (((F) \Leftrightarrow F) \land T) \\
&\land (F \Rightarrow (F)) \\
&\land (F \Rightarrow F)
\end{aligned}
$$

Example - Graph Library





$$
\begin{aligned}
\varphi &= T \\
&\wedge (T \Leftrightarrow T) \\
&\wedge (F \Rightarrow T) \\
&\wedge (F \Rightarrow T) \\
&\wedge ((T \Leftrightarrow T) \wedge \textcolor{red}{\mathbf{F}}) \\
&\wedge (F \Leftrightarrow F) \\
&\wedge ((F \Leftrightarrow F) \wedge T) \\
&\wedge (F \Rightarrow F) \\
&\wedge (F \Rightarrow T)
\end{aligned}
$$

Selection:
{GraphLibrary, EdgeType, Directed, Undirected}

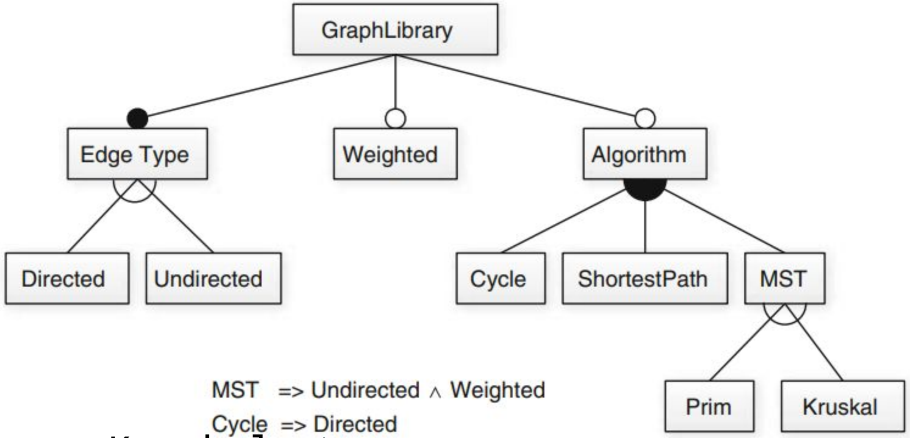
Consistent Feature Models

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

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 $\Rightarrow$ Weighted
2. ψ = Prim $\Rightarrow$ Weighted
3. ψ = $\neg$(Prim $\wedge$ Kruskal)
4. ψ = Weighted $\Rightarrow$ MST

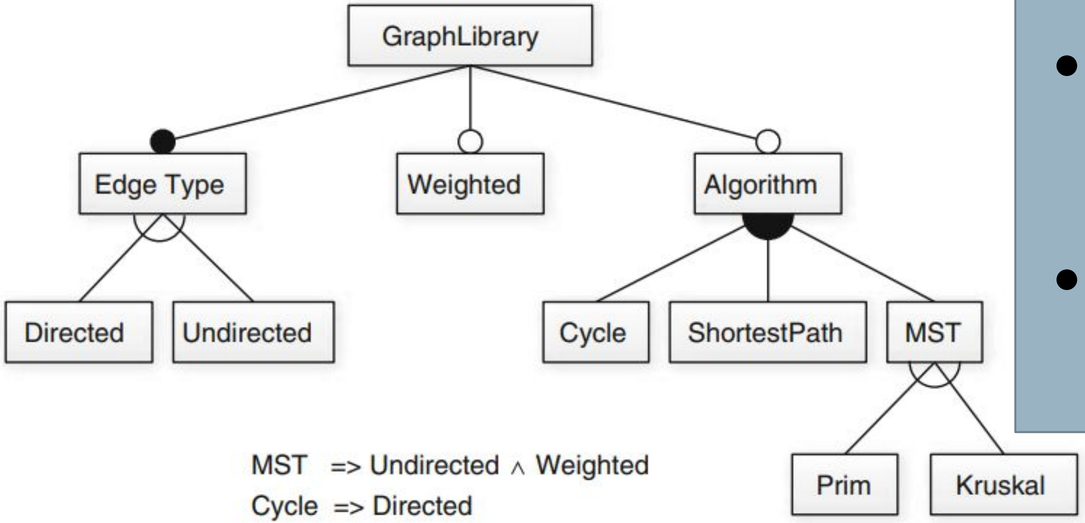
For each, assess whether ($\varphi \wedge \neg\psi$) is satisfiable.

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

Dead and Mandatory Features

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

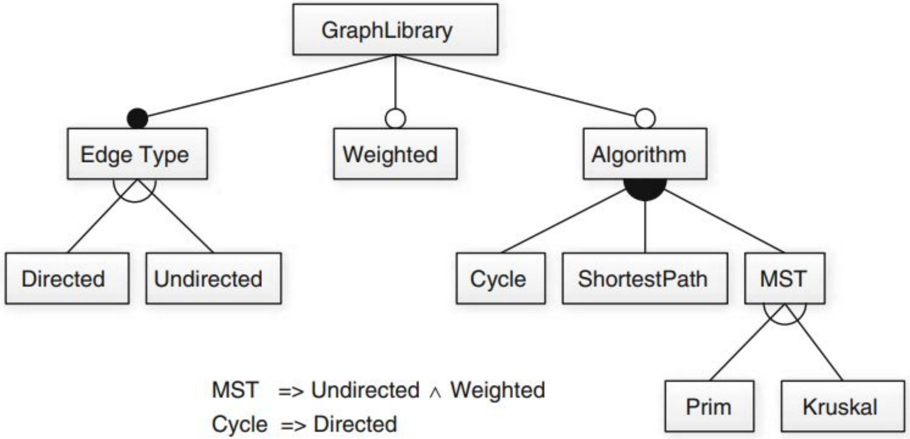
Example - Graph Library



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

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

Example - Graph Library

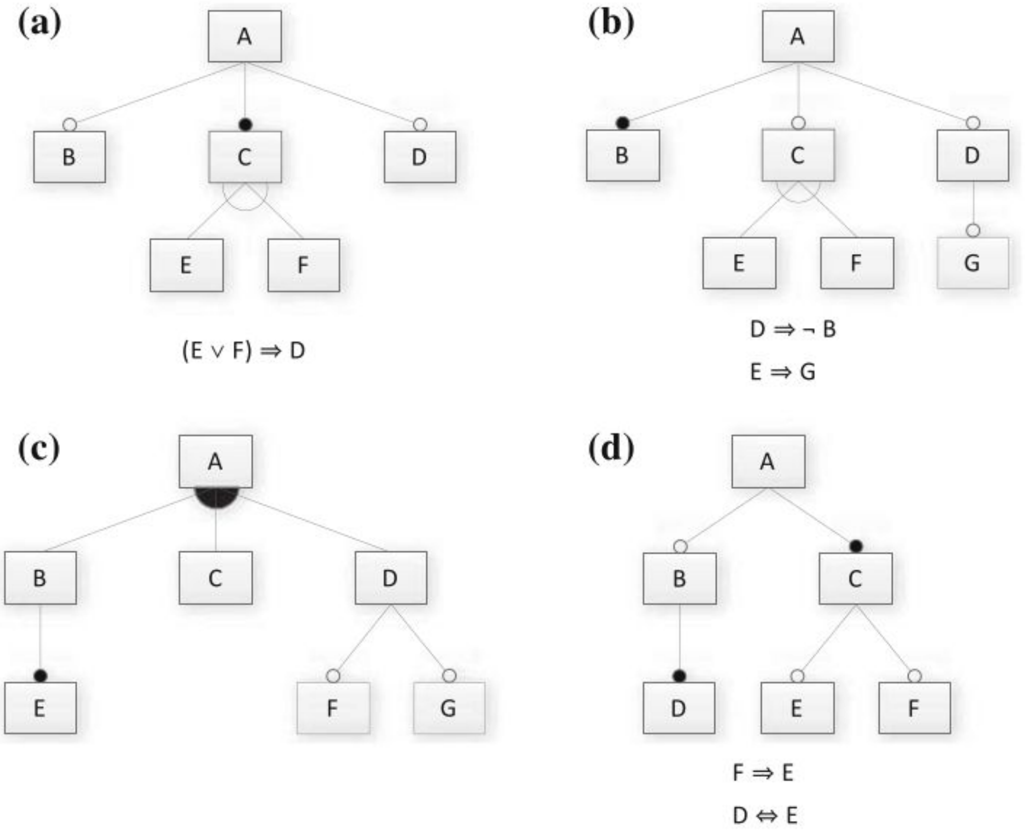


$$
\begin{aligned}
\varphi = {} & \mathit{GraphLibrary} \\
& \land (\mathit{EdgeType} \Leftrightarrow \mathit{GraphLibrary}) \\
& \land (\mathit{Weighted} \Rightarrow \mathit{GraphLibrary}) \\
& \land (\mathit{Algorithm} \Rightarrow \mathit{GraphLibrary}) \\
& \land (((\mathit{Directed} \lor \mathit{Undirected}) \Leftrightarrow \mathit{EdgeType}) \land \neg(\mathit{Directed} \land \mathit{Undirected})) \\
& \land ((\mathit{Cycle} \lor \mathit{ShortestPath} \lor \mathit{MST}) \Leftrightarrow \mathit{Algorithm}) \\
& \land (((\mathit{Prim} \lor \mathit{Kruskal}) \Leftrightarrow \mathit{MST}) \land \neg(\mathit{Prim} \land \mathit{Kruskal})) \\
& \land (\mathit{MST} \Rightarrow (\mathit{Undirected} \land \mathit{Weighted})) \\
& \land (\mathit{Cycle} \Rightarrow \mathit{Directed})
\end{aligned}
$$

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

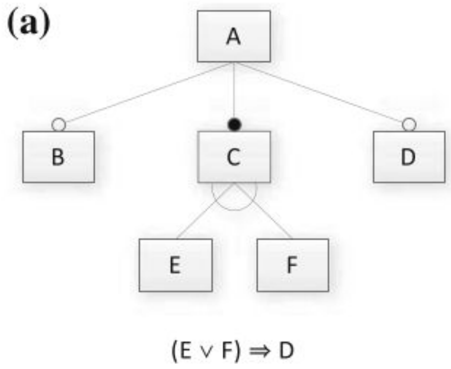
Activity

- 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?



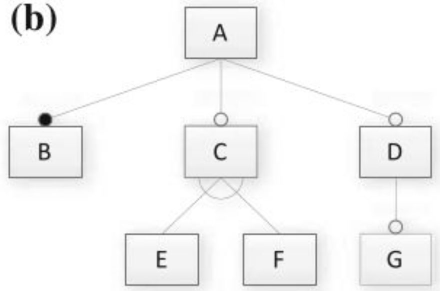
A ∧ (B ⇒ A) ∧ (C ⇔ A) ∧ (D ⇒ A) ∧
((C ⇔ (E ∨ F)) ∧ ¬(E ∧ F)) ∧ ((E ∨ F) ⇒ D))

- 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?

(b)



$A \wedge (B \Leftrightarrow A) \wedge (C \Rightarrow A) \wedge (D \Rightarrow A) \wedge$
$((C \Leftrightarrow (E \vee F)) \wedge \neg(E \wedge F)) \wedge (G \Rightarrow D) \wedge (D \Rightarrow \neg B)$
$\wedge$
$(E \Rightarrow G)$

- 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).

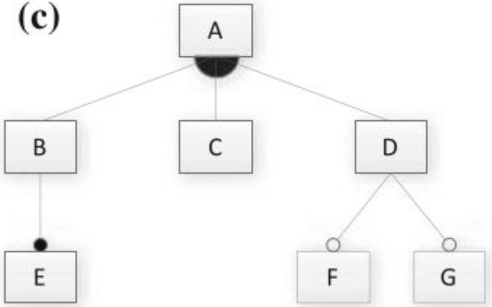
Solution (C)

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

$A \wedge ((B \vee C \vee D) \Leftrightarrow A) \wedge (E \Leftrightarrow B) \wedge (F \Rightarrow D) \wedge (G \Rightarrow D)$

- 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)



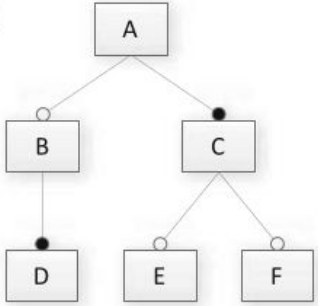
Solution (D)

- 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)



$F \Rightarrow E$

$D \Leftrightarrow E$

We Have Learned

- A product is a **valid** selection of features.
- Feature models capture the constraints that define whether a selection is valid.
  - Feature diagrams represent feature relationships visually.
  - Propositional logic represents feature relationships as formulae that can be used in analyses.

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).
  - Can add propositions to check facts about model (e.g., dead or mandatory features)

Next Time

- Project Workshop
  - Robocode
  - Installing FeatureIDE, Eclipse
  - GitLab setup
- Assignment 1 - due November 13
- Assignment 2 up soon.
  - Domain modelling!

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY