CHALMERS
UNIVERSITY OF TECHNOLOGY

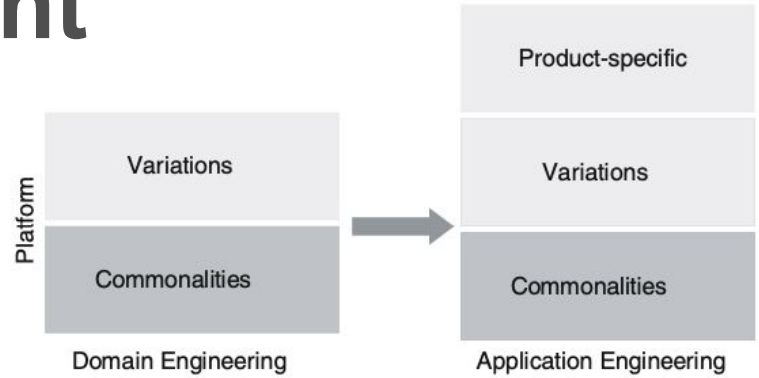
UNIVERSITY OF GOTHENBURG

Lecture 3: Feature Modeling

Gregory Gay
TDA 594/DIT 593 - November 9, 2021

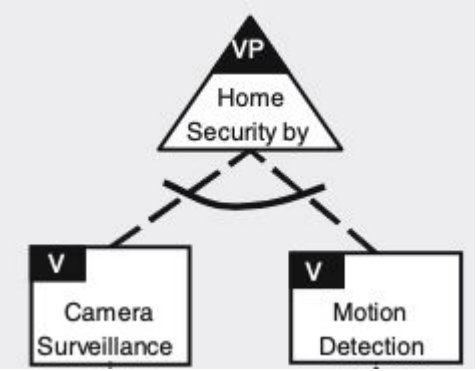
Variability Management

- Commonality
  - Shared between all products.
  - Implemented in core platform.
- Variability
  - Shared by subset of products.
  - Implemented in core platform, enabled in subset.
- Product-specific
  - Unique to a single product.
  - Platform must support unique adaptations.



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

Constraints on Variability

- Variability Dependencies
  - Dependencies between features at one variation point.
  - How many features can we choose for this point?
  - Which are mandatory? Optional?
- Feature Dependencies
  - Dependencies between features at same or different variation points.
  - Choosing one feature requires choosing or excluding another feature.

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.

Today's Goals

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

Feature Diagrams

Features and Feature Dependencies

- Generally a functionality of the software.
- Can be **mandatory** or **optional**.
- Features are connected by their **relationships**.
  - Selecting A *allows* B to be selected.
  - Selecting A *requires* B to be selected.
    - **Variation Point:** Selecting A requires selecting one of (B, C, D).
- A feature model describes these relationships.

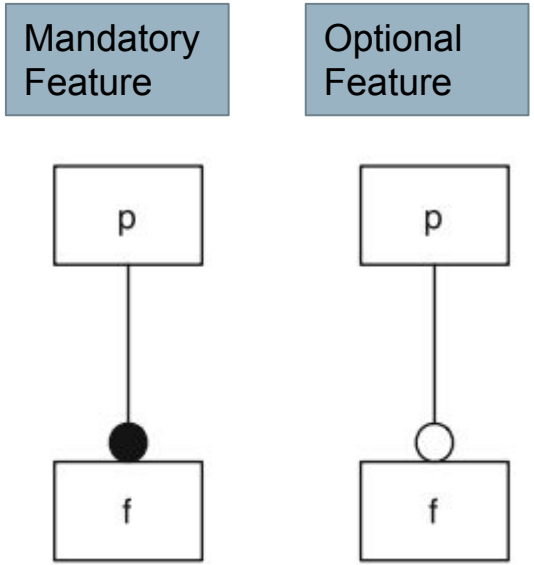
Identifying Features

- Aspects of the domain reflected in the software.
  - Externally-visible functions of software.
  - Aspects of non-functional behavior that can be controlled.
    - “Precision Mode” vs “Battery-Preserving Mode”
- Must represent a **distinct** and **well-understood** aspect of the system.

Understanding a Feature

- To model a feature, consider:
  - Description and requirements
  - Relationship to other features
    - (hierarchy, ordering, grouping)
  - External dependencies (hardware, software)
  - Configuration knowledge (activated by default?)
  - Constraints (requires feature X, excludes Y)
  - Effect on non-functional properties
  - Attributes (number, parameters)
  - Potential feature interactions.

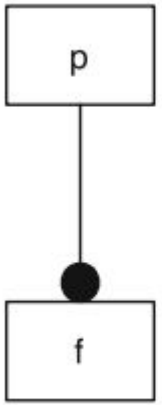
Feature Diagrams



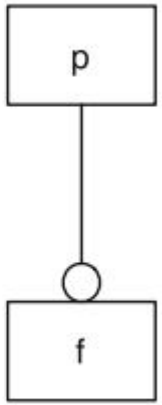
- Tree where nodes represent features.
- Shows parent-child relationship.
  - F can only be selected when P is selected.
  - Parent tends to be more general, child is more specific.
    - Parent - Sensor, Child - RADAR

Feature Diagrams

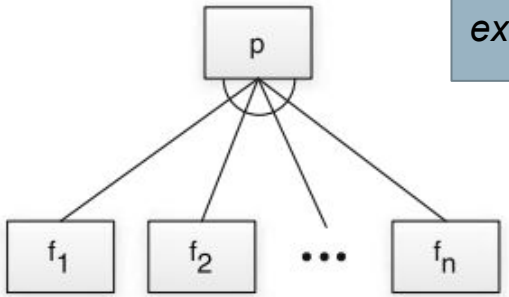
Mandatory Feature



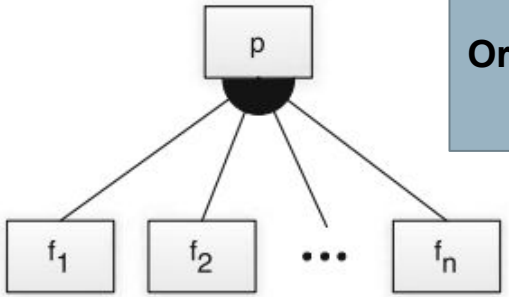
Optional Feature



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



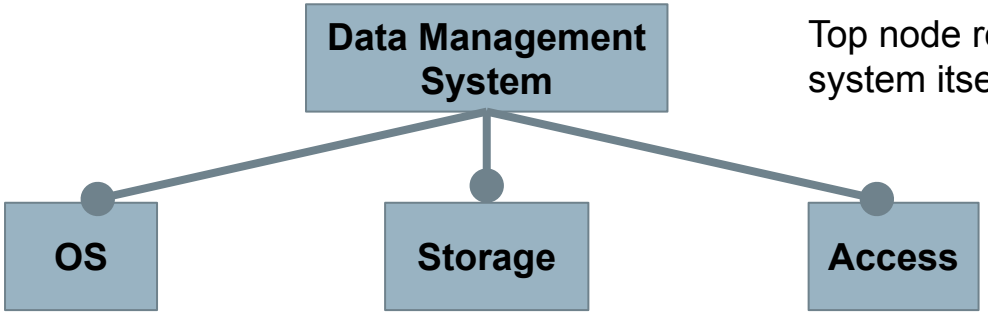
Or: Choose **at least** one



Cross-Tree Constraints

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

Example - Data Management

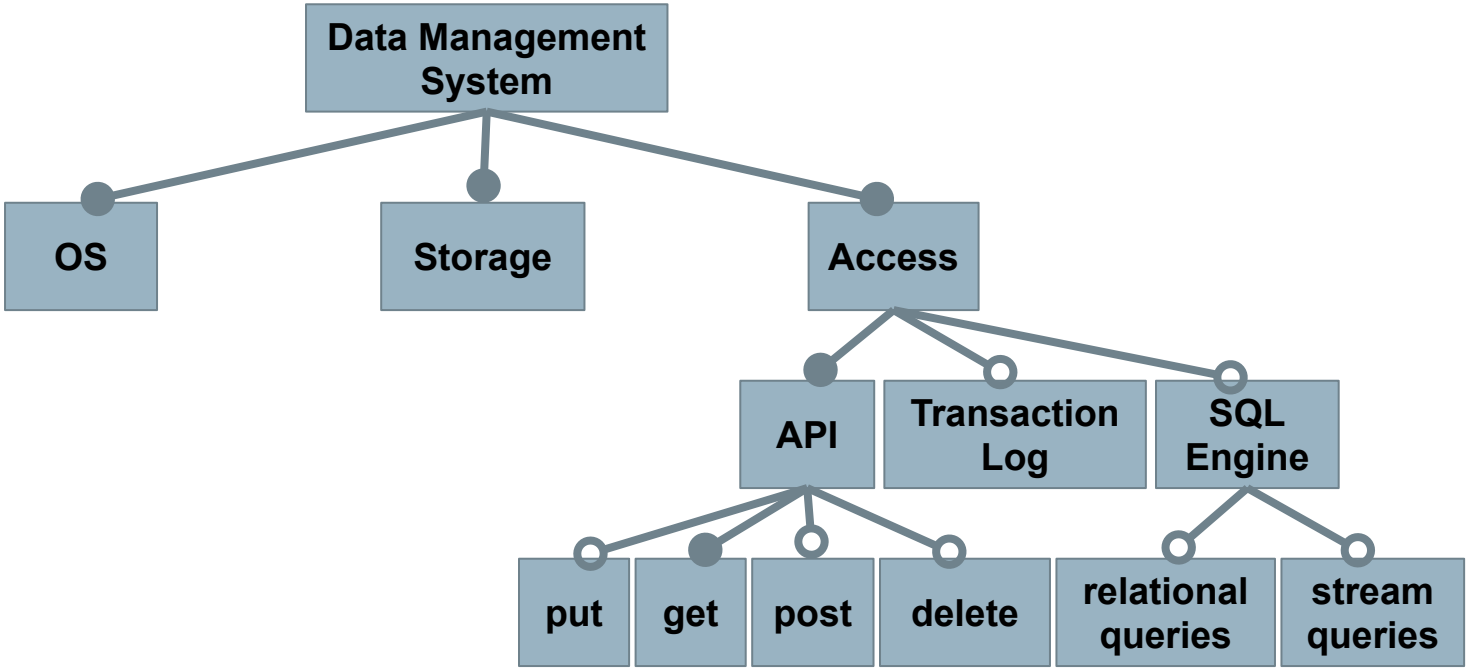


Top node represents the system itself.

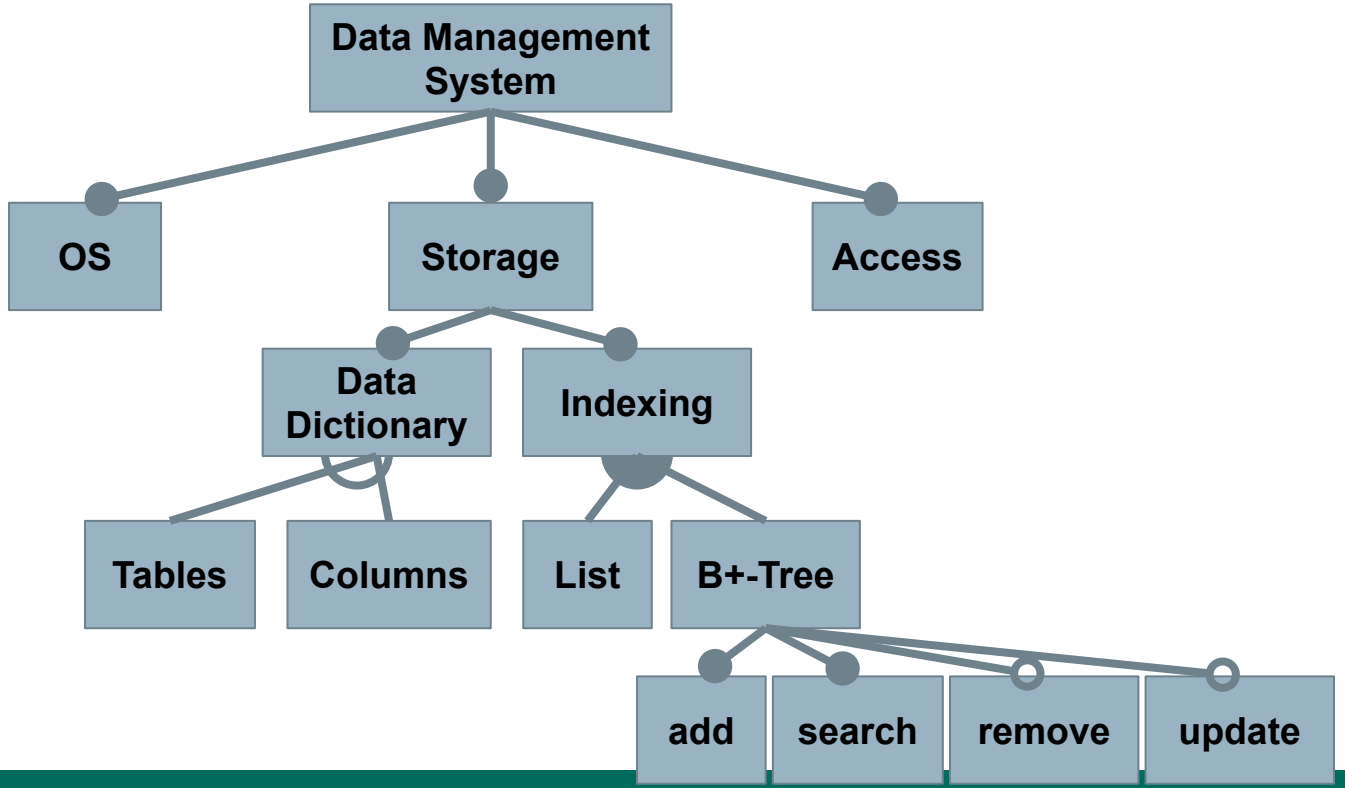
Hierarchy goes from general/abstract to specific.

First layer represents “types” of functionality.

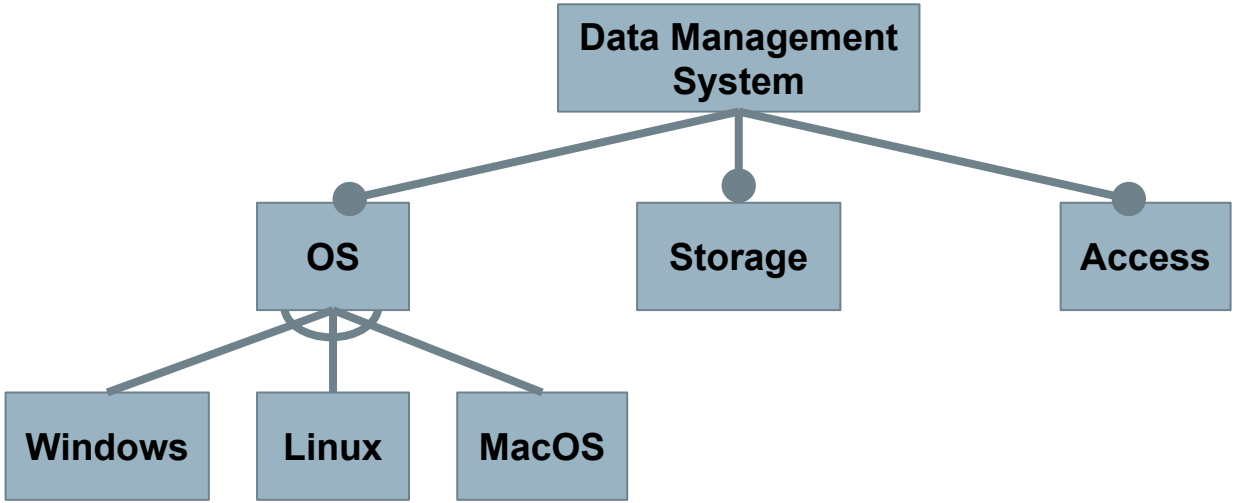
Example - Data Management



Example - Data Management

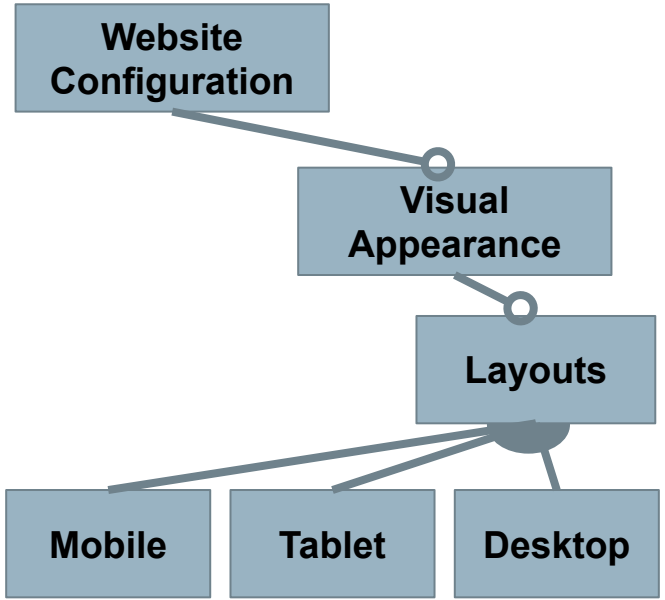


Example - Data Management

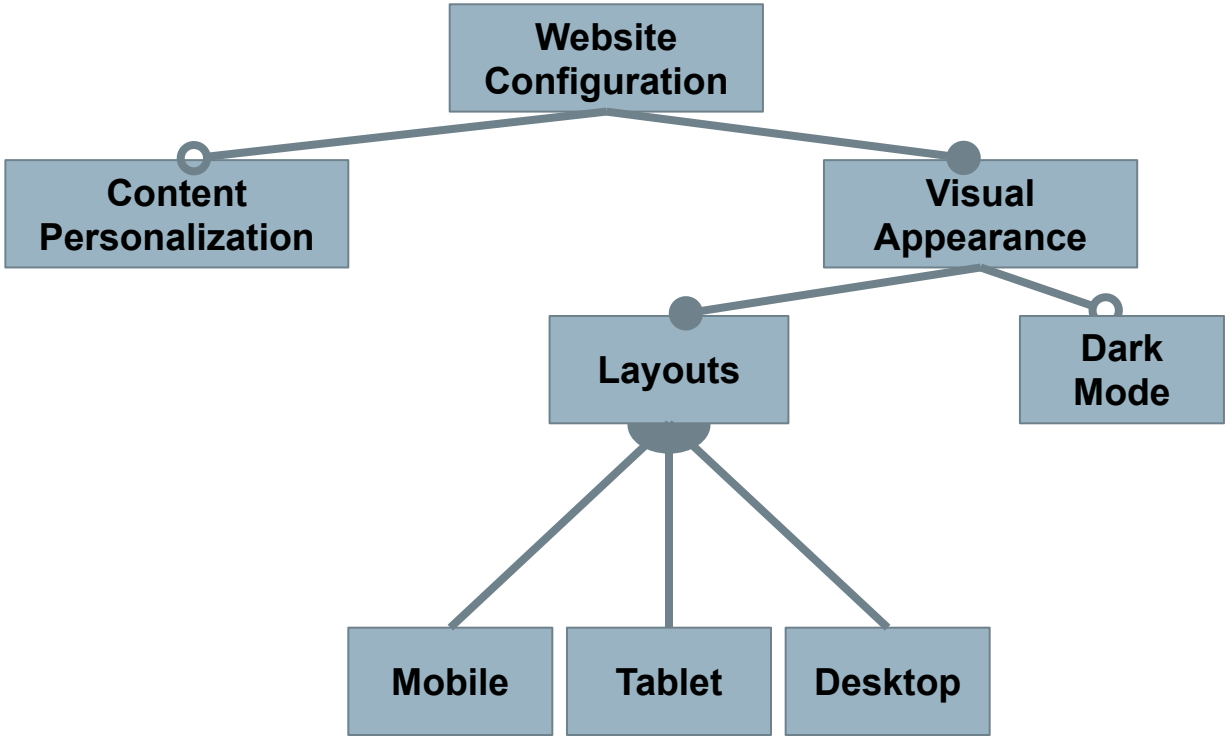


Example - Website Configuration

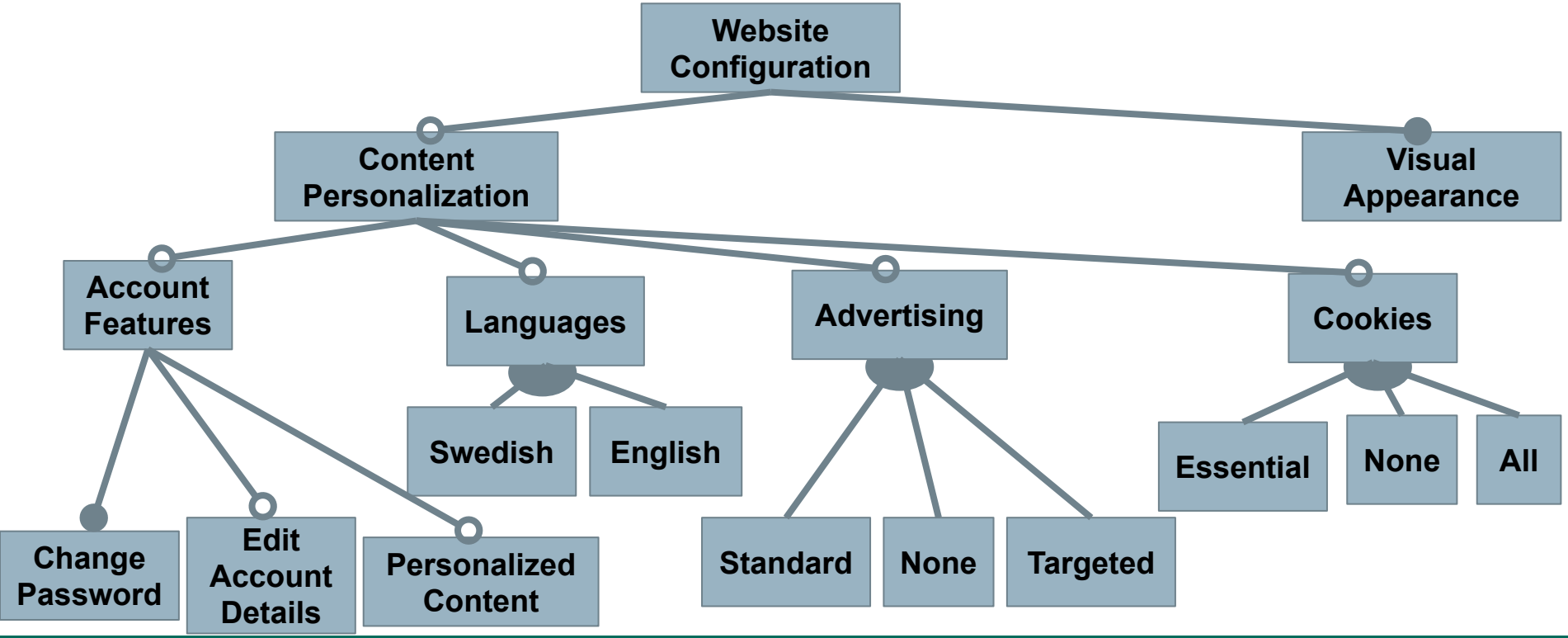
- SPL that provides website functionality.
- One feature - adjusts layout based on the device.
- What other aspect of the site could be features?
  - Consider visual appearance and personalized content.



Example - Website Configuration



Example - Website Configuration



Let’s take a break!

Activity - Smart TV OS

https://bit.ly/3H0wF2k

- 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

https://bit.ly/3H0wF2k

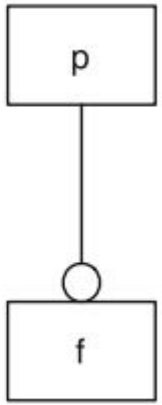
- Which features will many (or few) customers want?
- Which features might distinguish your product from others on the market?
- Don’t try to capture all features, but an interesting subset (aim for 15-25).
- Capture dependencies between features using visual structures and cross-tree constraints.

https://bit.ly/3H0wF2k

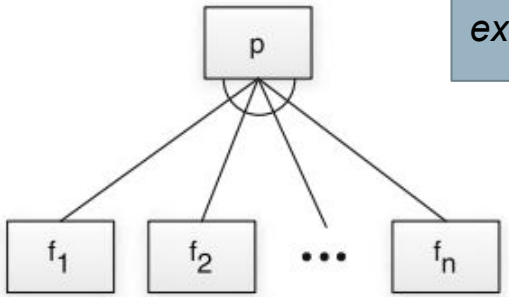
Review - Feature Diagrams

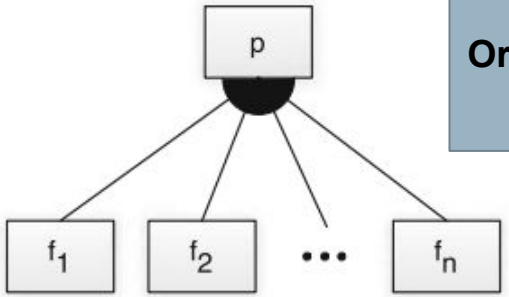
Mandatory Feature

Optional Feature



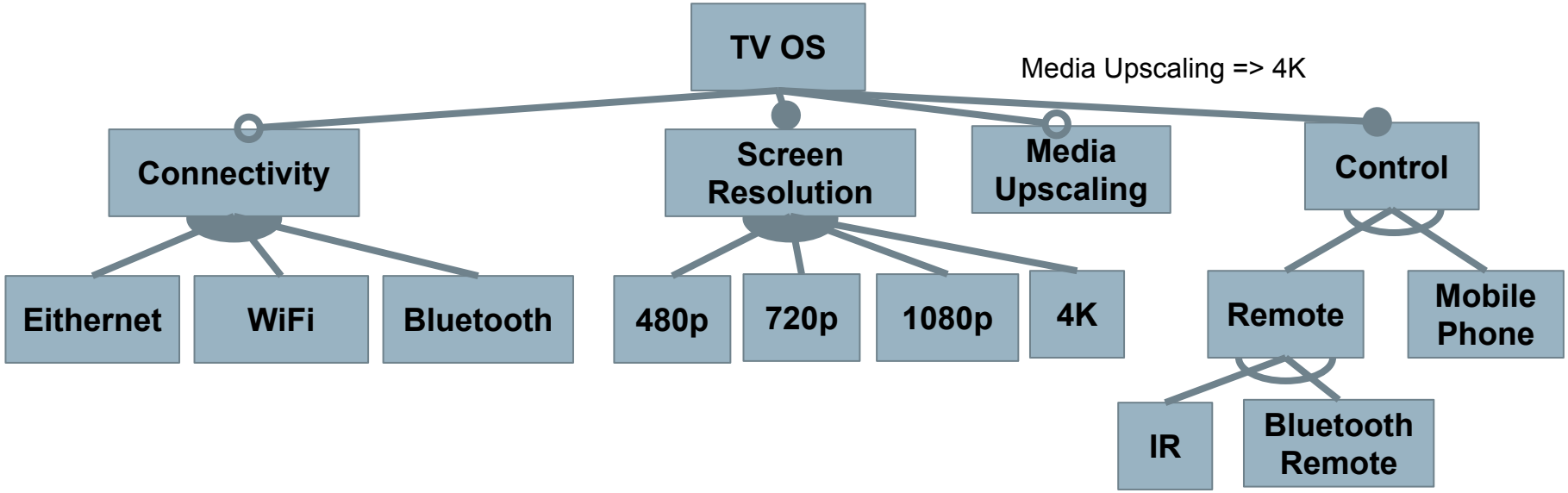
Alternative (mutually exclusive choice): Choose *exactly* one





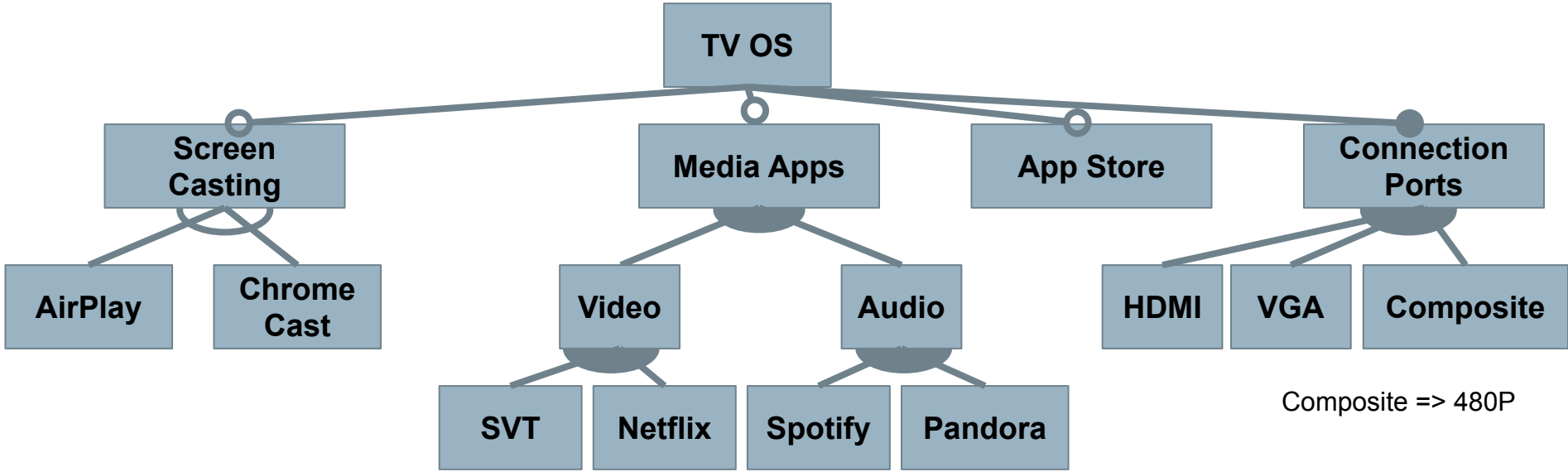
Or: Choose **at least** one

Possible Solution



Bluetooth Remote => Bluetooth

Possible Solution



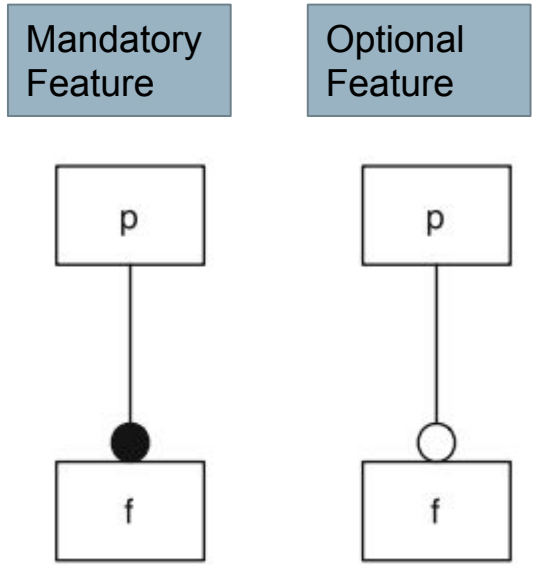
Composite => 480P

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

Propositional Logic and Feature Model Analysis

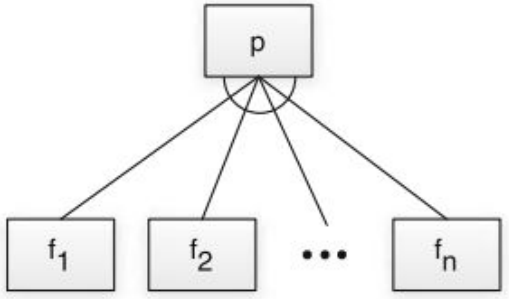
Propositional Logic

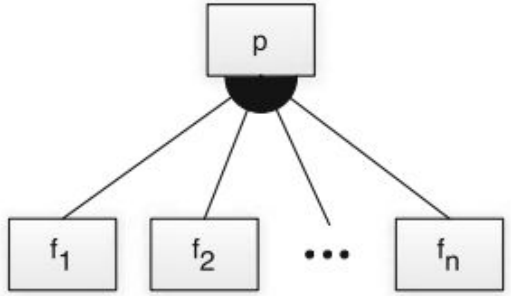
- **Mandatory:** If parent is selected, the child must be.
  - `mandatory(p, f) ≡ f ⇔ p`
- **Optional:** Child may only be chosen if the parent is.
  - `optional(p, f) ≡ f ⇒ p`



Propositional Logic

- **Alternative:** Choose **exactly** one
  - $\text{alternative}(p, \{f_1,\ldots,f_n\}) \equiv ((f_1 \vee \ldots \vee f_n) \Leftrightarrow p) \wedge_{(fi,fj)} \neg(f_i \wedge f_j)$
- **Or:** Choose **at least** one
  - $\text{or}(p, \{f_1,\ldots,f_n\}) \equiv ((f_1 \vee \ldots \vee f_n) \Leftrightarrow p)$

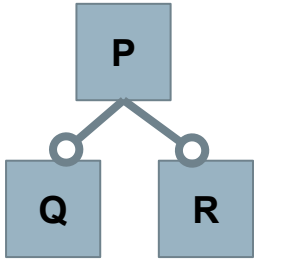




Propositional Logic

- 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 (∧)

$$\varphi = P \wedge (Q \Rightarrow P) \wedge (R \Rightarrow P) \wedge (Q \Rightarrow R)$$



Q => R

Variability-Aware Analysis

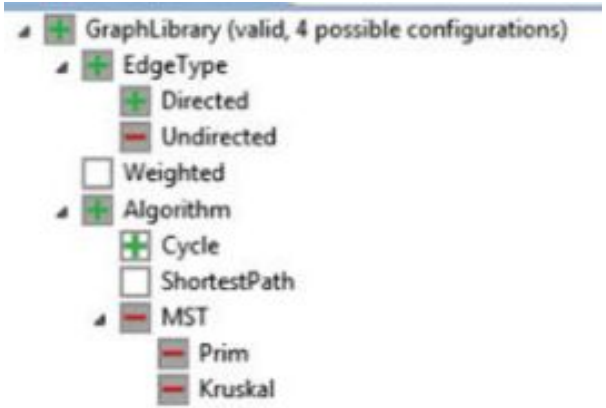
- Verification techniques do not extend to SPLs.
  - More product variations than atoms in the universe.
- Sometimes, can restrict to subset (HP printers).
- **Variability-Aware Analyses** can examine whole product line (or reasonable subset).

Analyses of Feature Models

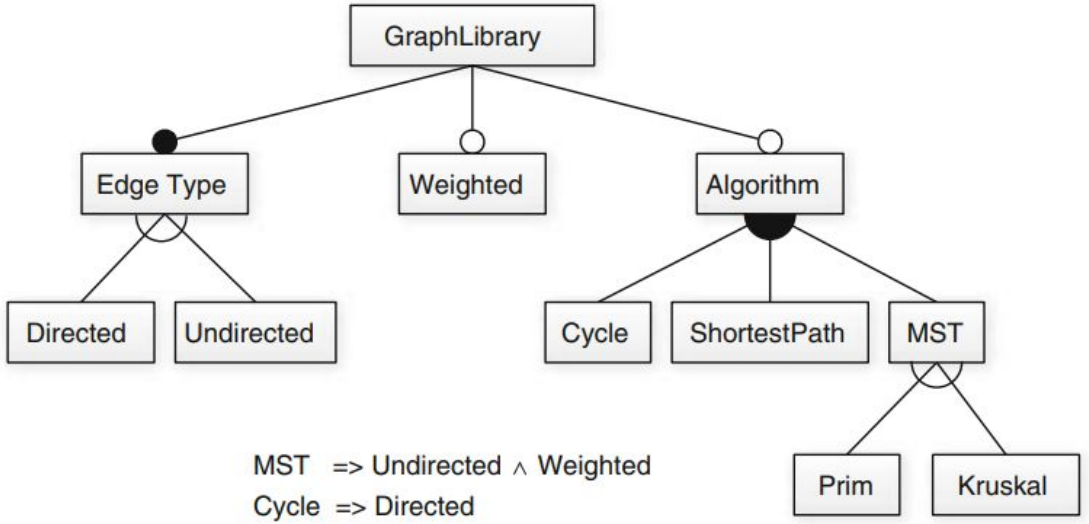
- Is a feature selection valid?
- Is the feature model consistent?
- Do our assumptions hold (testing)?
- Which features are mandatory?
- Which features can never be selected (dead)?
- How many valid selections does model have?
- Are two models equivalent?
- Given partial selection, what must be included?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

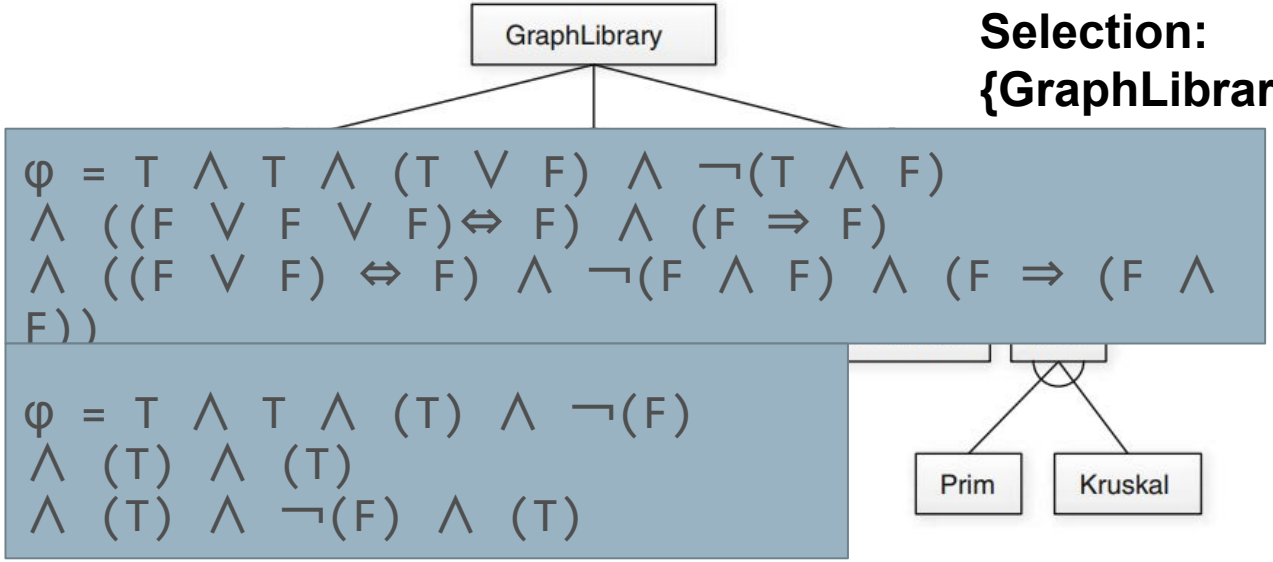


Example - Graph Library

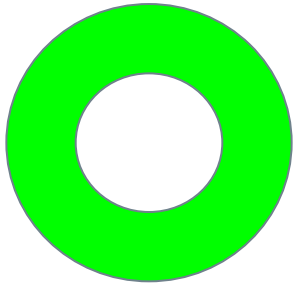


$$\phi = \texttt{GraphLibrary} \wedge \texttt{EdgeType} \wedge (\texttt{Directed} \vee \texttt{Undirected}) \wedge \neg(\texttt{Directed} \wedge \texttt{Undirected})$$
$$\wedge ((\texttt{Cycle} \vee \texttt{ShortestPath} \vee \texttt{MST}) \Leftrightarrow \texttt{Algorithm}) \wedge (\texttt{Cycle} \Rightarrow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

Example - Graph Library

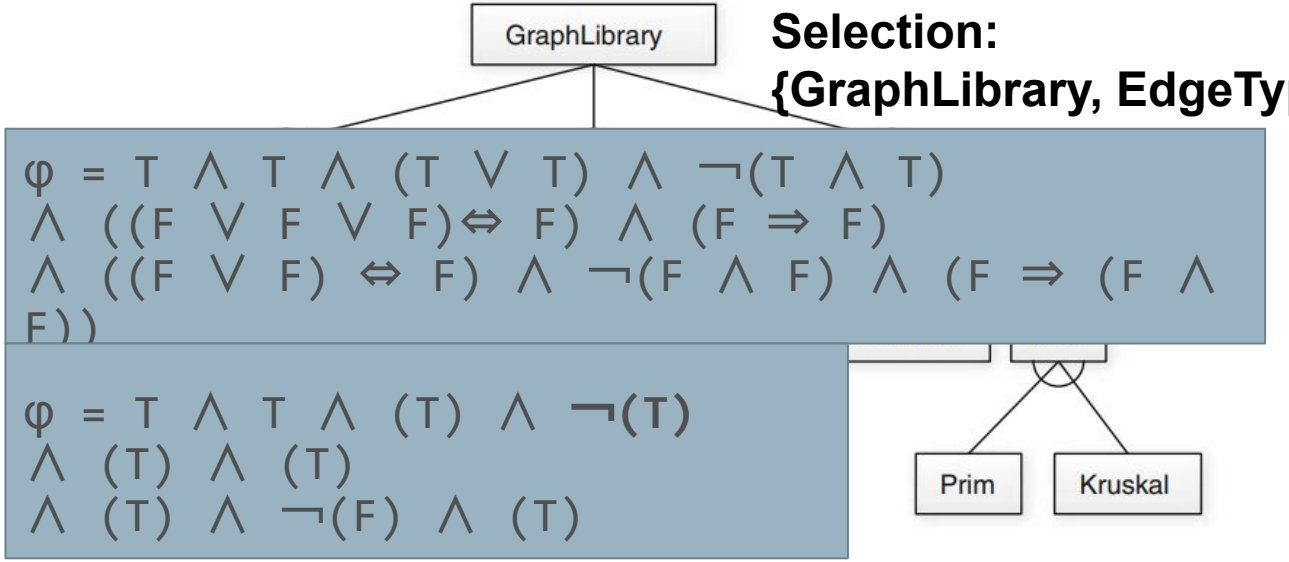


Selection:
{GraphLibrary, EdgeType, Directed}



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

Example - Graph Library



Selection:
{GraphLibrary, EdgeType, Directed, Undirected}

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

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.

Next Time

- Analysis of feature models
  - Video lecture - up on Canvas

- Assignment 1 - due November 14
  - Make sure you get approval from supervisor for case study subject.

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY