

Propositional Logic

Overview

Formal logic

- A **formal language** in which (well-formed) sentences are expressed
- A **semantics** that defines the meaning of the language expressions
- A **proof system** that provides rules for deriving valid judgements

Propositional logic

- Propositional Logic (PL) deals with **propositions** and their relationships
- **Propositions** can be either true or false
- Atomic formulas are **propositional variables** that represent propositions
- Compound formulas are built with **logical connectives**
- Each formula is either true or false for a given variable interpretation

Syntax

- Propositional variables
 - $A, B, C, \dots$
- Logical connectives
 - $\top$ (true), $\perp$ (false), $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies), $\leftrightarrow$ (equivalent)
- Auxiliary symbols
 - Parenthesis (,)
- Well-formed formulas follow the syntactic rules of propositional logic

Syntax

$$A, B, C, \dots \in \mathcal{V}$$

$$\phi, \psi, \dots \in \mathbf{Form}_{\mathcal{V}}$$

$$\phi, \psi \doteq A, B, C, \dots$$

$$| \top$$

$$| \perp$$

$$| (\neg \phi)$$

$$| (\phi \wedge \psi)$$

$$| (\phi \vee \psi)$$

$$| (\phi \rightarrow \psi)$$

$$| (\phi \leftrightarrow \psi)$$

- If parenthesis are omitted, connectives have precedence $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, and are right-associative

Semantics

- A truth value **assignment** to variables $\mathcal{A} : \mathcal{V} \rightarrow \{0,1\}$
- Can be extended to $\mathcal{A} : \mathbf{Form}_{\mathcal{V}} \rightarrow \{0,1\}$ to compute the truth value of a formula ϕ
- If $\mathcal{A}(\phi) = 1$ we say that ϕ **holds** under $\mathcal{A}$, denoted by $\mathcal{A} \models \phi$
- If $\mathcal{A}(\phi) = 0$ we say that ϕ **does not hold** under $\mathcal{A}$, denoted by $\mathcal{A} \not\models \phi$

Truth table semantics

$\frac{\mathcal{A}(\top)}{1}$	$\frac{\mathcal{A}(\perp)}{0}$
-------------------------------	--------------------------------

$\mathcal{A}(\phi)$	$\mathcal{A}(\neg\phi)$
0	1
1	0

$\mathcal{A}(\phi)$	$\mathcal{A}(\psi)$	$\mathcal{A}(\phi \wedge \psi)$	$\mathcal{A}(\phi \vee \psi)$	$\mathcal{A}(\phi \rightarrow \psi)$	$\mathcal{A}(\phi \leftrightarrow \psi)$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Inductive semantics

$$\mathcal{A} \models \top$$

$$\mathcal{A} \not\models \perp$$

$$\mathcal{A} \models p \quad \text{iff} \quad \mathcal{A}(p) = 1$$

$$\mathcal{A} \models \neg \phi \quad \text{iff} \quad \mathcal{A} \not\models \phi$$

$$\mathcal{A} \models \phi \wedge \psi \quad \text{iff} \quad \mathcal{A} \models \phi \text{ and } \mathcal{A} \models \psi$$

$$\mathcal{A} \models \phi \vee \psi \quad \text{iff} \quad \mathcal{A} \models \phi \text{ or } \mathcal{A} \models \psi$$

$$\mathcal{A} \models \phi \rightarrow \psi \quad \text{iff} \quad \mathcal{A} \not\models \phi \text{ or } \mathcal{A} \models \psi$$

$$\mathcal{A} \models \phi \leftrightarrow \psi \quad \text{iff} \quad \mathcal{A} \models \phi \text{ iff } \mathcal{A} \models \psi$$

Terminology

- A formula ϕ is
 - **valid** or a **tautology** iff it holds under *all* assignments, and we write $\models \phi$
 - **satisfiable** iff it holds under *some* assignments
 - **unsatisfiable** or a **contradiction** iff it does *not* hold under *all* assignments
 - **refutable** iff it does *not* hold under *some* assignment
- A formula ϕ is valid iff $\neg\phi$ is unsatisfiable

Examples

- $B \vee (A \rightarrow \neg B)$ is valid

A	B	$\neg B$	$A \rightarrow \neg B$	$B \vee (A \rightarrow \neg B)$
0	0	1	1	1
0	1	0	1	1
1	0	1	1	1
1	1	0	0	1

- $A \rightarrow (\neg A \vee B)$ is both satisfiable and refutable

A	B	$\neg A$	$\neg A \vee B$	$A \rightarrow (\neg A \vee B)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Consequence and equivalence

- ϕ is a **consequence** of ψ (or ψ **entails** ϕ), denoted by $\psi \models \phi$, if for every $\mathcal{A}$ that $\mathcal{A} \models \psi$, $\mathcal{A} \models \phi$ also holds
- ϕ and ψ are **equivalent**, denoted by $\psi \equiv \phi$, if $\phi \models \psi$ and $\psi \models \phi$
- $\phi \models \psi$ iff $\models \phi \rightarrow \psi$ and $\phi \equiv \psi$ iff $\models \phi \leftrightarrow \psi$

Basic equivalences

$$\phi \vee \phi \equiv \phi$$

$$\phi \vee \neg\phi \equiv \top$$

$$\phi \vee \top \equiv \top$$

$$\phi \vee \perp \equiv \phi$$

$$\phi \vee \psi \equiv \psi \vee \phi$$

$$\phi \vee (\phi \wedge \delta) \equiv (\phi \vee \phi) \wedge (\phi \vee \delta)$$

$$\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$

$$\phi \wedge (\phi \vee \psi) \equiv \phi$$

$$\phi \rightarrow \psi \equiv \neg\phi \vee \psi$$

$$\phi \wedge \phi \equiv \phi$$

$$\phi \wedge \neg\phi \equiv \perp$$

$$\phi \wedge \top \equiv \phi$$

$$\phi \wedge \perp \equiv \perp$$

$$\phi \wedge \psi \equiv \psi \wedge \phi$$

$$\phi \wedge (\phi \vee \delta) \equiv (\phi \wedge \phi) \vee (\phi \wedge \delta)$$

$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$$

$$\neg\neg\phi \equiv \phi$$

$$\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$$

Decidability

- A decision problem is **decidable** if there exists a mechanical method (aka an algorithm) for determining if it is either true or false
- The decision problem “Is ϕ satisfiable?”, also known as the **Boolean satisfiability problem** or **SAT**, is decidable
 - Naïve approach, exponential in the number of variables: enumerate all possible assignments and build the truth table for ϕ
 - Later we will see that modern **SAT solvers** implement more sophisticated methods...
- Hence the decision problem “Is ϕ valid?” is also decidable



Modelling with PL

Formal modelling

- Logic allows us to reason formally during various software development tasks:
 - Program analysis
 - Artificial intelligence
 - Hardware verification
 - Programming languages
- (The relationship is symbiotic: software provides implementations of logics)
- The system under analysis must be represented in the selected formal system
 - We call this task **formal modelling**

Allocation problems

Allocation problem

- Decide if a set of “items” can be placed in a set of “containers”
- Subject to a set of generic (often implicit) constraints
 - At least / at most one item per container
 - At least / at most one container per item
- And a set of problem specific constraints
 - A specific item does not want to be placed in a specific container
 - Two specific items do not want to be placed together
 - ...

Allocating in PL

- Declare a matrix of $| \text{Items} | \times | \text{Containers} |$ propositional variables
 - Variable $p_{x,a}$ is true iff item x is allocated to container a
- Specify the allocation constraints with propositional formulas
- SAT solve to get allocation

Generic constraints

- Consider 3 containers (1,2,3) and 2 items (a , b)

- At least one container per item

$$(p_{a,1} \vee p_{a,2} \vee p_{a,3}) \wedge (p_{b,1} \vee p_{b,2} \vee p_{b,3})$$

- At most one container per item

		Container		
		1	2	3
Item	a	$p_{a,1}$	$p_{a,2}$	$p_{a,3}$
	b	$p_{b,1}$	$p_{b,2}$	$p_{b,3}$

Generic constraints

- Consider 3 containers (1,2,3) and k items ($i_0, \dots, i_k$)

- At least one container per item

$$(p_{i_0,1} \vee p_{i_0,2} \vee p_{i_0,3}) \wedge \dots \wedge (p_{i_k,1} \vee p_{i_k,2} \vee p_{i_k,3})$$

- At most one container per item

		Container		
		1	2	3
Item	i_0	$p_{i_0,1}$	$p_{i_0,2}$	$p_{i_0,3}$
	i_1	$p_{i_1,1}$	$p_{i_1,2}$	$p_{i_1,3}$
	...	...	...	...
	i_k	$p_{i_k,1}$	$p_{i_k,2}$	$p_{i_k,3}$

Generic constraints

- Consider l containers $(c_0, \dots, c_l)$ and k items $(a_0, \dots, a_k)$

- At least one container per item

$$(p_{i_0, c_0} \vee \dots \vee p_{i_0, c_l}) \wedge \dots \wedge (p_{i_k, c_0} \vee \dots \vee p_{i_k, c_l})$$

- At most one container per item

		Container			
		c_0	c_1	...	c_l
Item	i_0	p_{i_0, c_0}	p_{i_0, c_1}	...	p_{i_0, c_l}
	i_1	p_{i_1, c_0}	p_{i_1, c_1}	...	p_{i_1, c_l}
	...	...	...	...	...
	i_k	p_{i_k, c_0}	p_{i_k, c_1}	...	p_{i_k, c_l}

Generic constraints

- Consider 3 containers (1,2,3) and 2 items (a , b)

- At least one container per item

$$(p_{a,1} \vee p_{a,2} \vee p_{a,3}) \wedge (p_{b,1} \vee p_{b,2} \vee p_{b,3})$$

- At most one container per item

$$\neg(p_{a,1} \wedge p_{a,2}) \wedge \neg(p_{a,1} \wedge p_{a,3}) \wedge \neg(p_{a,2} \wedge p_{a,3}) \\ \wedge \\ \neg(p_{b,1} \wedge p_{b,2}) \wedge \neg(p_{b,1} \wedge p_{b,3}) \wedge \neg(p_{b,2} \wedge p_{b,3})$$

		Container		
		1	2	3
Item	a	$p_{a,1}$	$p_{a,2}$	$p_{a,3}$
	b	$p_{b,1}$	$p_{b,2}$	$p_{b,3}$

Generic constraints

- Consider 3 containers (1,2,3) and 2 items (a , b)
- At least one container per item

$$(p_{a,1} \vee p_{a,2} \vee p_{a,3}) \wedge (p_{b,1} \vee p_{b,2} \vee p_{b,3})$$

- At most one container per item

$$\begin{aligned} & (\neg p_{a,1} \vee \neg p_{a,2}) \wedge (\neg p_{a,1} \vee \neg p_{a,3}) \wedge (\neg p_{a,2} \vee \neg p_{a,3}) \\ & \quad \wedge \\ & (\neg p_{b,1} \vee \neg p_{b,2}) \wedge (\neg p_{b,1} \vee \neg p_{b,3}) \wedge (\neg p_{b,2} \vee \neg p_{b,3}) \end{aligned}$$

		Container		
		1	2	3
Item	a	$p_{a,1}$	$p_{a,2}$	$p_{a,3}$
	b	$p_{b,1}$	$p_{b,2}$	$p_{b,3}$

Generic constraints

- Consider 3 containers (1,2,3) and k items ($i_0, \dots, i_k$)

- At least one container per item

$$(p_{i_0,1} \vee p_{i_0,2} \vee p_{i_0,3}) \wedge \dots \wedge (p_{i_k,1} \vee p_{i_k,2} \vee p_{i_k,3})$$

- At most one container per item

$$\begin{aligned} &(\neg p_{i_0,1} \vee \neg p_{i_0,2}) \wedge (\neg p_{i_0,1} \vee \neg p_{i_0,3}) \wedge (\neg p_{i_0,2} \vee \neg p_{i_0,3}) \\ &\quad \wedge \dots \wedge \\ &(\neg p_{i_k,1} \vee \neg p_{i_k,2}) \wedge (\neg p_{i_k,1} \vee \neg p_{i_k,3}) \wedge (\neg p_{i_k,2} \vee \neg p_{i_k,3}) \end{aligned}$$

		Container		
		1	2	3
Item	i_0	$p_{i_0,1}$	$p_{i_0,2}$	$p_{i_0,3}$
	i_1	$p_{i_1,1}$	$p_{i_1,2}$	$p_{i_1,3}$
	...	...	...	...
	i_k	$p_{i_k,1}$	$p_{i_k,2}$	$p_{i_k,3}$

Generic constraints

- Consider l containers $(c_0, \dots, c_l)$ and k items $(a_0, \dots, a_k)$
- At least one container per item

$$(p_{i_0, c_0} \vee \dots \vee p_{i_0, c_l}) \wedge \dots \wedge (p_{i_k, c_0} \vee \dots \vee p_{i_k, c_l})$$

- At most one container per item

$$\begin{aligned} & (\neg p_{i_0, c_0} \vee \neg p_{i_0, c_1}) \wedge (\neg p_{i_0, c_0} \vee \neg p_{i_0, c_2}) \wedge \dots \wedge (\neg p_{i_0, c_0} \vee \neg p_{i_0, c_l}) \\ & \quad \wedge \dots \wedge \\ & (\neg p_{i_0, c_x} \vee \neg p_{i_0, c_{x+1}}) \wedge (\neg p_{i_0, c_x} \vee \neg p_{i_0, c_{x+2}}) \wedge \dots \wedge (\neg p_{i_0, c_x} \vee \neg p_{i_0, c_l}) \\ & \quad \wedge \dots \wedge \\ & (\neg p_{i_k, c_x} \vee \neg p_{i_k, c_{x+1}}) \wedge (\neg p_{i_k, c_x} \vee \neg p_{i_k, c_{x+2}}) \wedge \dots \wedge (\neg p_{i_k, c_x} \vee \neg p_{i_k, c_l}) \end{aligned}$$

		Container			
		c_0	c_1	...	c_l
Item	i_0	p_{i_0, c_0}	p_{i_0, c_1}	...	p_{i_0, c_l}
	i_1	p_{i_1, c_0}	p_{i_1, c_1}	...	p_{i_1, c_l}
	...	...	...	...	...
	i_k	p_{i_k, c_0}	p_{i_k, c_1}	...	p_{i_k, c_l}

Checking assertions

- After adding the constraints describing the allocation problem we can check the validity of additional assertions
- To check if ϕ is valid add $\neg\phi$ as a constraint and check for satisfiability
 - If it is satisfiable, then $\neg\phi$ holds for some assignment, so ϕ is not valid

Placement of guests

- We have three chairs in a row and need to place Anne, Susan and Peter
 - Anne does not want to sit next to Peter
 - Anne does not want to sit in the left chair
 - Susan does not want to sit to the left of Peter
- Implicit generic constraints
 - Everyone must be sited in a chair (at least one chair per guest)
 - No more than one guest per chair (at most one guest per chair)

Variables

		Chair		
		Left	Center	Right
Guest	Anne	$x_{a,l}$	$x_{a,c}$	$x_{a,r}$
	Susan	$x_{s,l}$	$x_{s,c}$	$x_{s,r}$
	Peter	$x_{p,l}$	$x_{p,c}$	$x_{p,r}$

At least one chair per guest

$$(x_{a,l} \vee x_{a,c} \vee x_{a,r}) \wedge (x_{s,l} \vee x_{s,c} \vee x_{s,r}) \wedge (x_{p,l} \vee x_{p,c} \vee x_{p,r})$$

At most one guest per chair

$$\begin{aligned} & (\neg x_{a,l} \vee \neg x_{s,l}) \wedge (\neg x_{a,l} \vee \neg x_{p,l}) \wedge (\neg x_{s,l} \vee \neg x_{p,l}) \\ & \quad \wedge \\ & (\neg x_{a,c} \vee \neg x_{s,c}) \wedge (\neg x_{a,c} \vee \neg x_{p,c}) \wedge (\neg x_{s,c} \vee \neg x_{p,c}) \\ & \quad \wedge \\ & (\neg x_{a,r} \vee \neg x_{s,r}) \wedge (\neg x_{a,r} \vee \neg x_{p,r}) \wedge (\neg x_{s,r} \vee \neg x_{p,r}) \end{aligned}$$

Problem specific constraints

- Anne does not want to sit next to Peter

$$(x_{a,c} \rightarrow (\neg x_{p,l} \wedge \neg x_{p,r})) \wedge ((x_{a,l} \vee x_{a,r}) \rightarrow \neg x_{p,c})$$

- Anne does not want to sit in the left chair

$$\neg x_{a,l}$$

- Susan does not want to sit to the left of Peter

$$(x_{p,r} \rightarrow \neg x_{s,c}) \wedge (x_{p,c} \rightarrow \neg x_{s,l})$$

Finding a placement

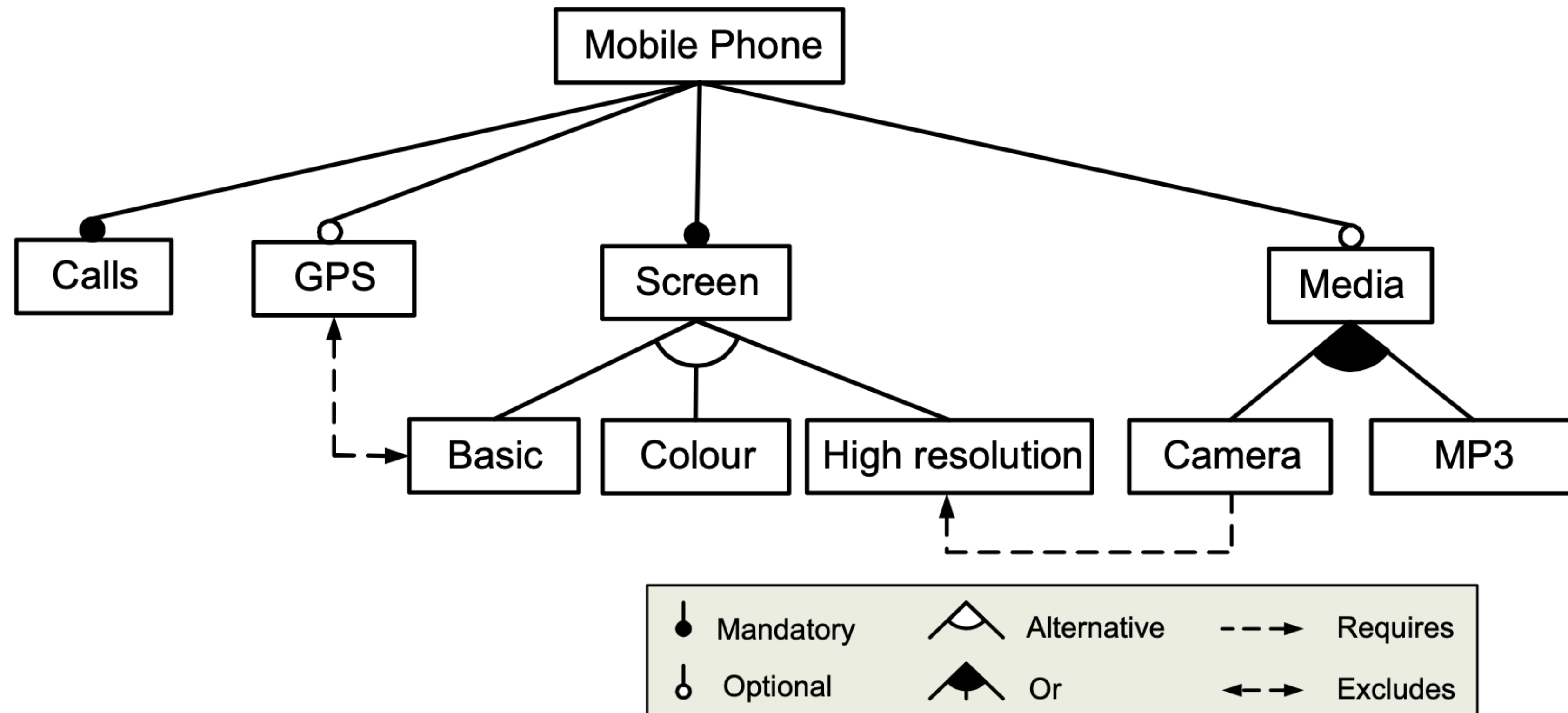
- A placement is an assignment to variables for which the constraints hold
 - SAT solving to show it is satisfiable: finding a $\mathcal{A}$ under which it holds
- We can do this manually, there are “only” $2^9 = 512$ possible assignments
 - Only one solution: $x_{p,l}, x_{a,c}, x_{s,r}$ true, all other variables false
- Later we will revisit this with automated SAT solvers

Feature model analysis

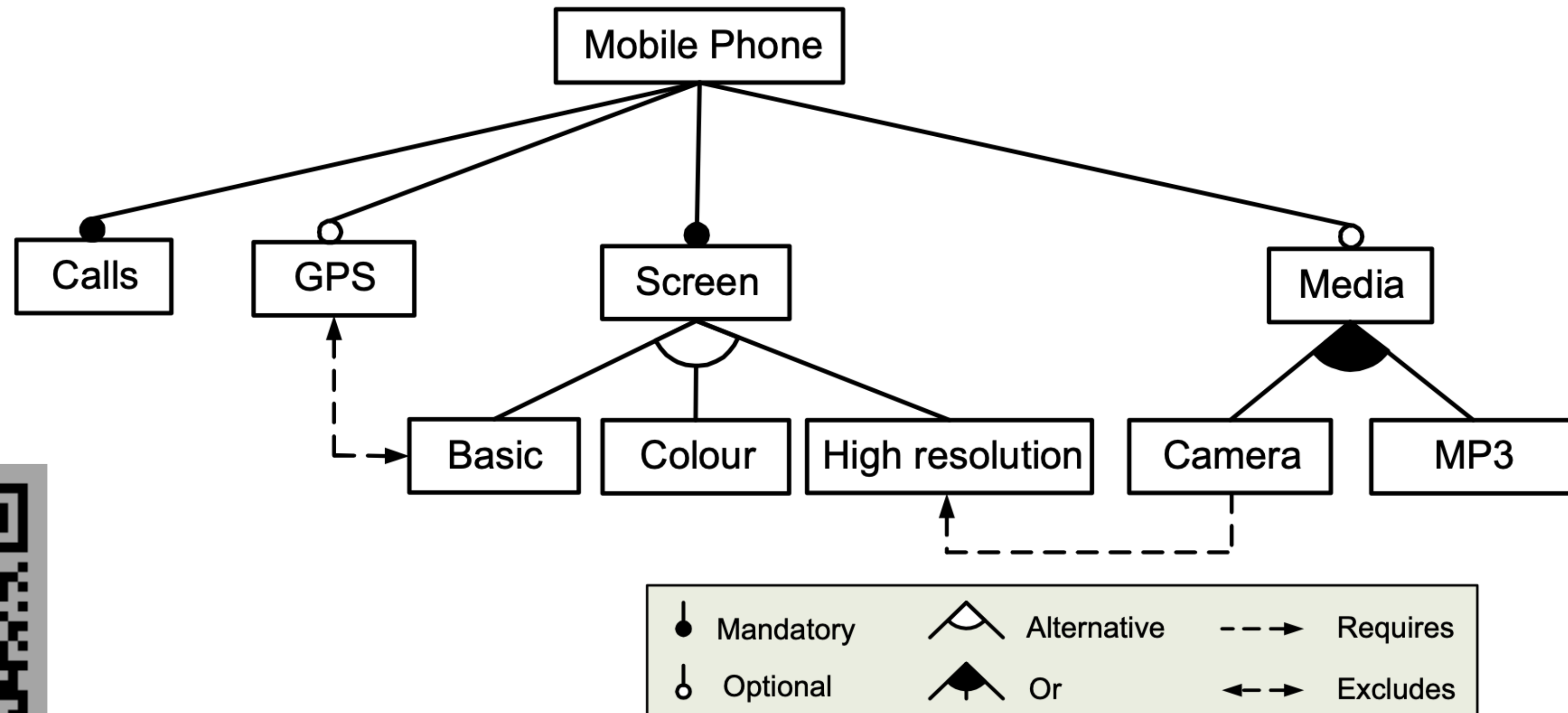
Variability modelling

- A **software product line** (SPL) is a family of software products
- Each **product** or **variant** supports different features
- A **feature** is an increment in program functionality
- A **feature model** is a compact representation of the variability of a SPL

Feature models



How many phone variants exist?



Feature model analysis

- Relevant analyses of a feature model
 - check if it is not void (there are products)
 - check if a feature is “dead” (no product can implement it)
 - check if a feature is “core” (all products implement it)
 - count how many different products exist
- All these analyses can be easily implemented using SAT solving

Feature model semantics

- A feature model can be encoded with a propositional formula ϕ
- The presence of each feature corresponds to a propositional variable
- Each feature model primitive corresponds to a conjunct of ϕ

Feature model semantics

r is the root feature

f_1 mandatory sub-feature of f

f_1 optional sub-feature of f

$f_1 \dots f_n$ or sub-features of f

$f_1 \dots f_n$ xor sub-features of f

f_1 requires f_2

f_1 excludes f_2

r

$$f_1 \leftrightarrow f$$

$$f_1 \rightarrow f$$

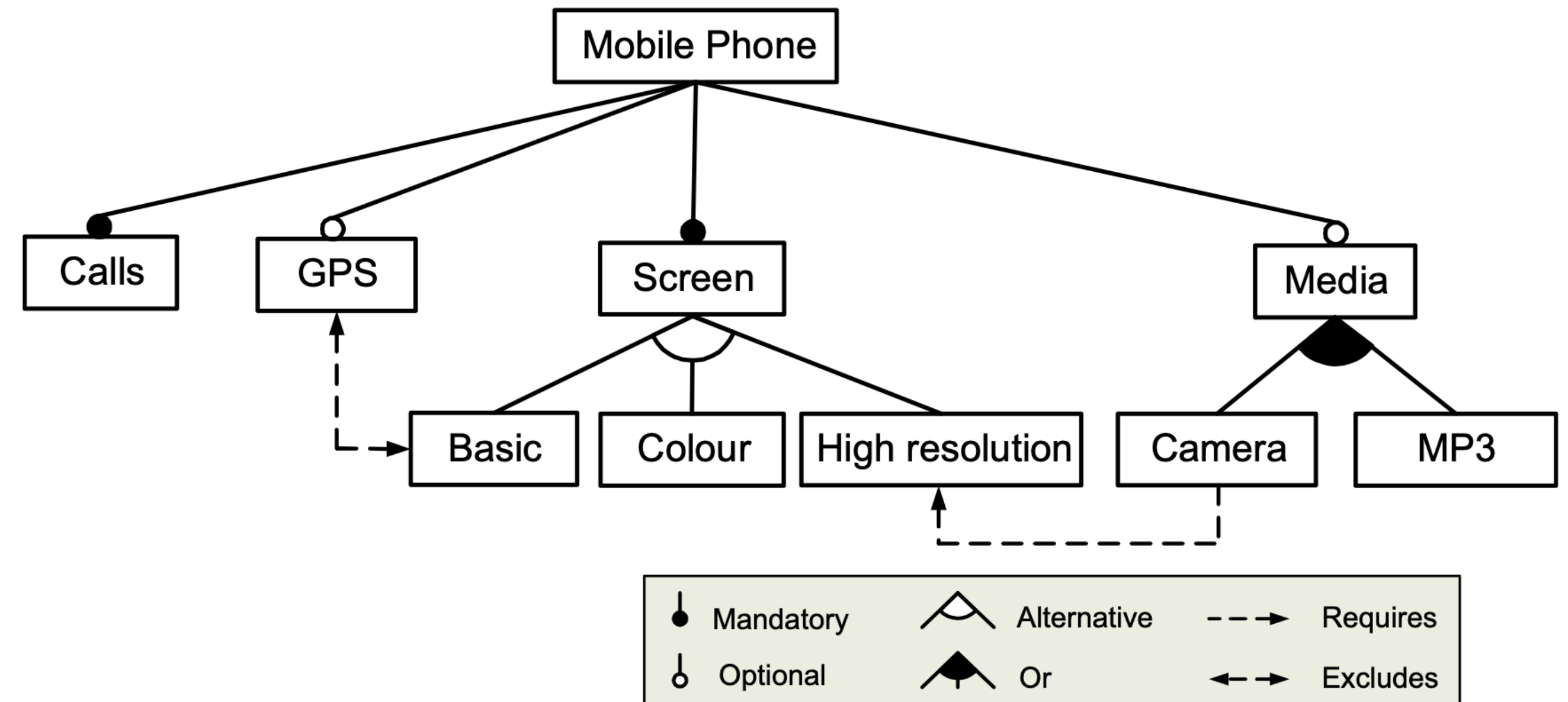
$$f_1 \vee \dots \vee f_n \leftrightarrow f$$

$$(f_1 \vee \dots \vee f_n \leftrightarrow f) \wedge \neg(f_i \wedge f_{i+1}) \wedge \dots \wedge \neg(f_i \wedge f_n)$$

$$f_1 \rightarrow f_2$$

$$\neg(f_1 \wedge f_2)$$

Feature model semantics



Phone

Calls $\leftrightarrow$ Phone

GPS $\rightarrow$ Phone

Screen $\leftrightarrow$ Phone

Media $\rightarrow$ Phone

$(\text{Basic} \vee \text{Color} \vee \text{Highres}) \leftrightarrow \text{Screen}$

$\neg(\text{Basic} \wedge \text{Color}) \wedge \neg(\text{Basic} \wedge \text{Highres}) \wedge \neg(\text{Color} \wedge \text{Highres})$

$(\text{Camera} \vee \text{MP3}) \leftrightarrow \text{Media}$

$\neg(\text{GPS} \wedge \text{Basic})$

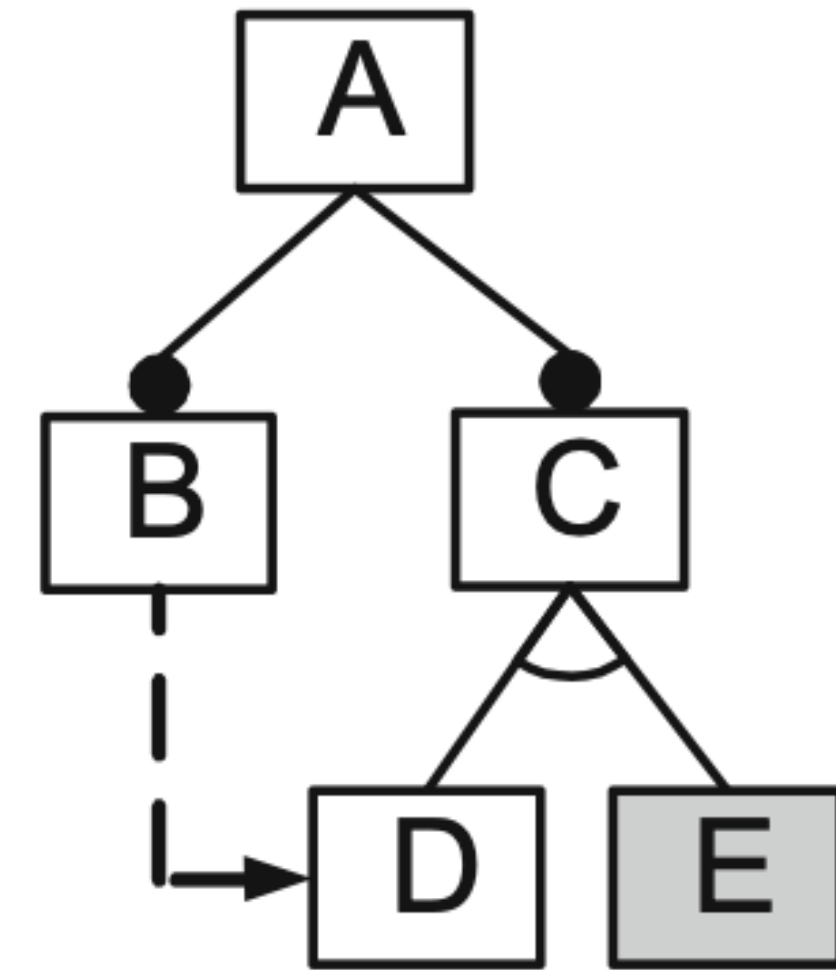
Camera $\rightarrow$ Highres

Non voidness

- Given a formula ϕ that encodes the semantics of a feature model
- To check if the feature model is not void
 - check if ϕ is satisfiable

Dead feature

- Given a formula ϕ that encodes the semantics of a feature model
- To check if feature f is dead
 - check if $\phi \wedge f$ is unsatisfiable



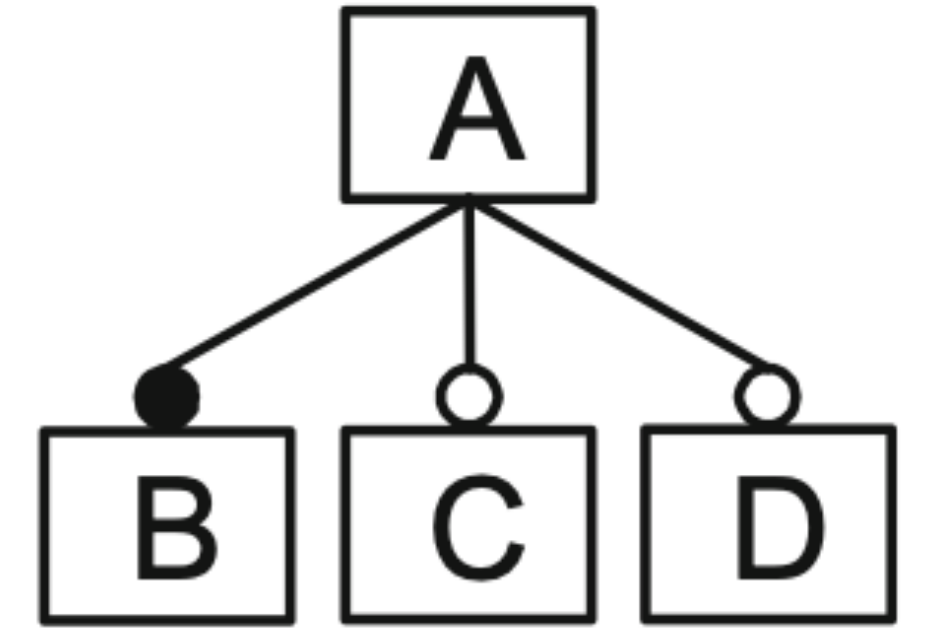
Core feature

- Given a formula ϕ that encodes the semantics of a feature model
- To check if feature f is core
 - check if $\phi \wedge \neg f$ is unsatisfiable

Counting products

- Given a formula ϕ that encodes the semantics of a feature model
- To count the number of products count the number of iterations of the following cycle
 - Repeat while ϕ is satisfiable
 - extract an assignment $\mathcal{A}$ under which ϕ holds
 - convert $\mathcal{A}$ to a formula $\overline{\mathcal{A}}$ that holds only for $\mathcal{A}$
 - add $\neg \overline{\mathcal{A}}$ as a new conjunct of ϕ

Counting products



$$\phi_0 \equiv A \wedge (A \leftrightarrow B) \wedge (C \rightarrow A) \wedge (D \rightarrow A)$$

$$\mathcal{A}_1 \equiv \{A \mapsto 1, B \mapsto 1, C \mapsto 0, D \mapsto 0\}$$

$$\phi_1 \equiv A \wedge (A \leftrightarrow B) \wedge (C \rightarrow A) \wedge (D \rightarrow A) \wedge \neg(A \wedge B \wedge \neg C \wedge \neg D)$$

$$\mathcal{A}_2 \equiv \{A \mapsto 1, B \mapsto 1, C \mapsto 1, D \mapsto 0\}$$

$$\phi_2 \equiv A \wedge (A \leftrightarrow B) \wedge (C \rightarrow A) \wedge (D \rightarrow A) \wedge \neg(A \wedge B \wedge \neg C \wedge \neg D) \wedge \neg(A \wedge B \wedge C \wedge \neg D)$$

$$\mathcal{A}_3 \equiv \{A \mapsto 1, B \mapsto 1, C \mapsto 0, D \mapsto 1\}$$

$$\phi_3 \equiv \dots$$

$$\mathcal{A}_4 \equiv \{A \mapsto 1, B \mapsto 1, C \mapsto 1, D \mapsto 1\}$$

$$\phi_4 \equiv \dots$$