CS477 Formal Software Dev Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

Slides based in part on previous lectures
by Mahesh Vishwanathan, and by Gul Agha

January 31, 2018

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?
  - Is P a tautology?

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?
  - Is P a tautology?
    - P is true in all valuations

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?
  - Is P a tautology?
    - P is true in all valuations
  - Note: A general algorithm to answer the last can be used to answer the second and vice versa.

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?
  - Is P a tautology?
    - P is true in all valuations
  - Note: A general algorithm to answer the last can be used to answer the second and vice versa.
- Difficulty: Answering if P is satisfiable is NP-complete

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?
  - Is P a tautology?
    - P is true in all valuations
  - Note: A general algorithm to answer the last can be used to answer the second and vice versa.
- Difficulty: Answering if P is satisfiable is NP-complete
- Algorithms exist with good performance in general practice

Model Checking for Propositions

- Problem: Would like an efficient way to answer for a given proposition P:
  - Does a given valuation satisfy P?
    - Valuation gives specific values for variables in P
  - Is P satisfiable?
    - Does there exist a valuation that makes P true?
  - Is P a tautology?
    - P is true in all valuations
  - Note: A general algorithm to answer the last can be used to answer the second and vice versa.
- Difficulty: Answering if P is satisfiable is NP-complete
- Algorithms exist with good performance in general practice
- BDDs are one such

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
    - Leaves are labeled by true or false.

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
  - Leaves are labeled by true or false.
  - Leaves have no out edges

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
    - Leaves are labeled by true or false.
    - Leaves have no out edges
    - Internal nodes are labeled by atomic propositions (variables)

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
  - Leaves are labeled by true or false.
  - Leaves have no out edges
  - Internal nodes are labeled by atomic propositions (variables)
  - Internal nodes have exactly two out edges

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
  - Leaves are labeled by true or false.
  - Leaves have no out edges
  - Internal nodes are labeled by atomic propositions (variables)
  - Internal nodes have exactly two out edges
  - Left edges labeled false and right edges labeled true.

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
  - Leaves are labeled by true or false.
  - Leaves have no out edges
  - Internal nodes are labeled by atomic propositions (variables)
  - Internal nodes have exactly two out edges
  - Left edges labeled false and right edges labeled true.
    - Think 0 and 1

Binary Decision Trees

- Binary decision tree is a (rooted, directected) edge and vertex labeled tree with two types of verices – internal nodes, and leaves – such that:
  - Leaves are labeled by true or false.
  - Leaves have no out edges
  - Internal nodes are labeled by atomic propositions (variables)
  - Internal nodes have exactly two out edges
  - Left edges labeled false and right edges labeled true.
    - Think 0 and 1
  - For each path (branch) in the tree, each atomic proposition may label at most one vertex of that path.

Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node

Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node
  - Any variable not on path can have any value

Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node
  - Any variable not on path can have any value
  - Leaf label says whether a valuation assigning those values to those variables

Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node
  - Any variable not on path can have any value
  - Leaf label says whether a valuation assigning those values to those variables
    - Is a model (true, the tree accepts the valuation)

Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node
  - Any variable not on path can have any value
  - Leaf label says whether a valuation assigning those values to those variables
    - Is a model (true, the tree accepts the valuation)
    - Or not a model (false, the tree rejects the valuation)

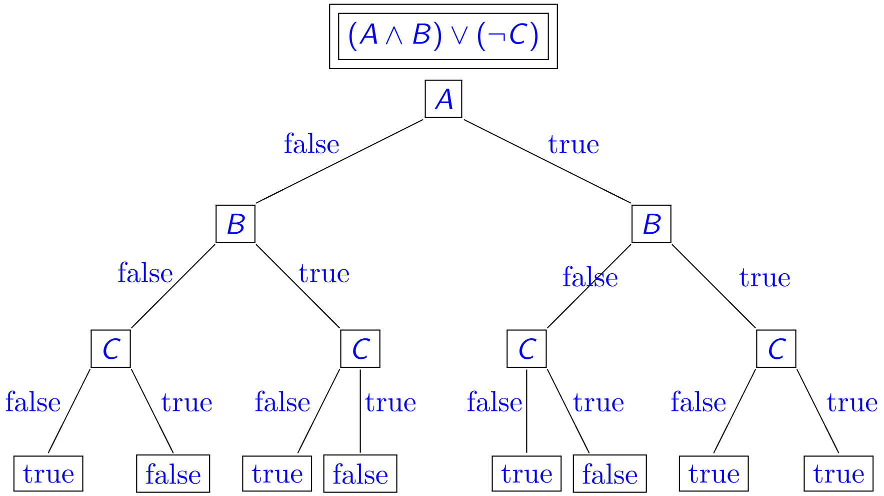
Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node
  - Any variable not on path can have any value
  - Leaf label says whether a valuation assigning those values to those variables
    - Is a model (true, the tree accepts the valuation)
    - Or not a model (false, the tree rejects the valuation)
  - Each valuation matches exactly one branch

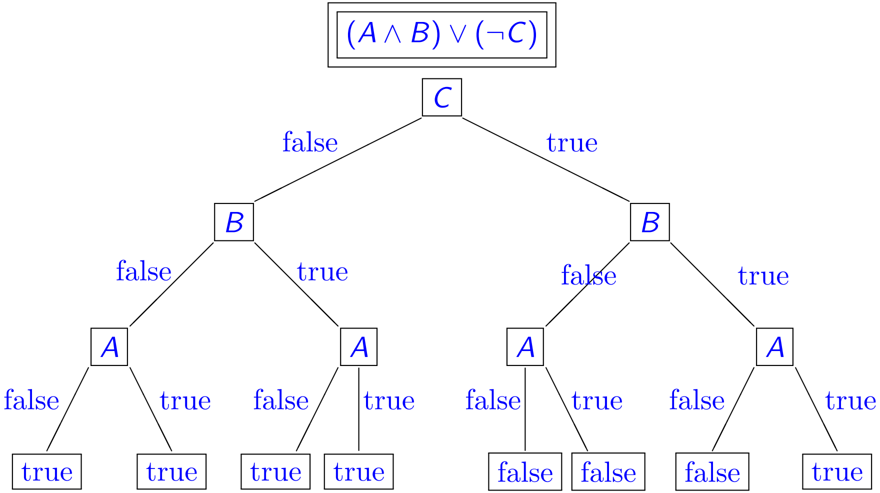
Binary Decision Trees

- Binary decision trees can record the set of all models (and non-models) of a proposition
  - Path records a valuation: out edge label gives value for variable labeling an internal node
  - Any variable not on path can have any value
  - Leaf label says whether a valuation assigning those values to those variables
    - Is a model (true, the tree accepts the valuation)
    - Or not a model (false, the tree rejects the valuation)
  - Each valuation matches exactly one branch
  - More than one valuation may (will) match a given branch

Example:

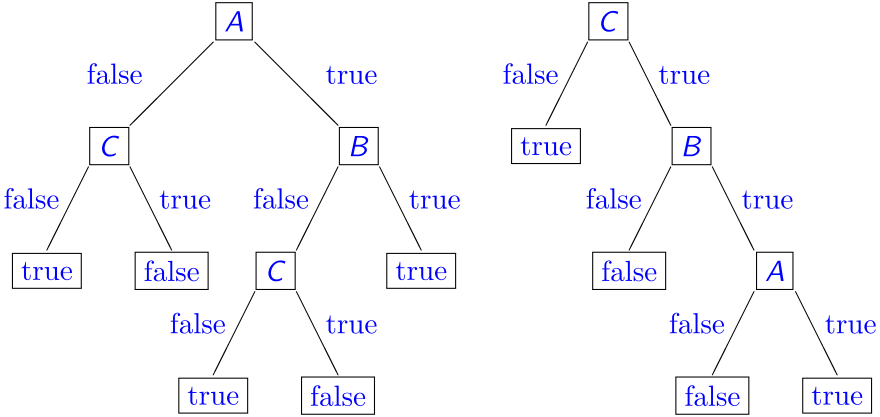

$(A \wedge B) \vee (\neg C)$
A
false
true
B
B
false
true
false
true
C
C
C
C
false
true
false
true
false
true
false
true
true
false
true
false
true
false
true
true

Example: Different Variable Ordering - Different Tree



Example: Many Logically Equivalent Trees

$$\boxed{\boxed{(A \wedge B) \vee (\neg C)}}$$



Alternate Syntax for Propositional Logic

- Still have constants $\{\mathbf{T}, \mathbf{F}\}$
- Still have countable set AP of propositional variables a.k.a. atomic propositions
- Only one ternary connective: the conditional if_then_else_
  - First argument only a variable
  - Second and third arguments propositions
  - Example

$$\textit{if } C \textit{ then if } B \textit{ then if } A \textit{ then } \mathbf{T} \textit{ else } \mathbf{F} \textit{ else } \mathbf{F} \textit{ else } \mathbf{T}$$

  - Represents the last tree above

Semantics for Conditional Propositional Logic

- Define when a valuation v satisfies a conditional proposition by

$$\begin{array}{l} v \models \mathbf{T} \\ v \not\models \mathbf{F} \\ v \models \mathit{if}\ A\ \mathit{then}\ P_t\ \mathit{else}\ P_f \text{ iff} \\ \quad v(A) = \mathrm{true} \text{ and } v \models P_t \text{ or} \\ \quad v(A) = \mathrm{false} \text{ and } v \models P_f \end{array}$$

- Example: let $v = \{A \mapsto \mathrm{true},\ B \mapsto \mathrm{true}, C \mapsto \mathrm{true}\}$

$$v \models \mathit{if}\ C\ \mathit{then}\ \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{T}$$

Semantics for Conditional Propositional Logic

- Define when a valuation v satisfies a conditional proposition by

$$
\begin{array}{l}
v \models \mathbf{T} \\
v \not\models \mathbf{F} \\
v \models \mathit{if}\ A\ \mathit{then}\ P_t\ \mathit{else}\ P_f \text{ iff} \\
\qquad v(A) = \mathrm{true} \text{ and } v \models P_t \text{ or} \\
\qquad v(A) = \mathrm{false} \text{ and } v \models P_f
\end{array}
$$

- Example: let $v = \{A \mapsto \mathrm{true},\ B \mapsto \mathrm{true}, C \mapsto \mathrm{true}\}$

  $v \models \mathit{if}\ C\ \mathit{then}\ \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{T}$ since
  $v(C) = \mathrm{true}$ and
  $v \models \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}$

Semantics for Conditional Propositional Logic

- Define when a valuation v satisfies a conditional proposition by

$$\begin{aligned} &v \models \mathbf{T} \\ &v \not\models \mathbf{F} \\ &v \models \mathit{if}\ A\ \mathit{then}\ P_t\ \mathit{else}\ P_f \text{ iff} \\ &\qquad v(A) = \mathrm{true} \text{ and } v \models P_t \text{ or} \\ &\qquad v(A) = \mathrm{false} \text{ and } v \models P_f \end{aligned}$$

- Example: let $v = \{A \mapsto \mathrm{true},\ B \mapsto \mathrm{true}, C \mapsto \mathrm{true}\}$

$v \models \mathit{if}\ C\ \mathit{then}\ \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{T}$ since
$v(C) = \mathrm{true}$ and
$v \models \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}$ since
$v(B) = \mathrm{true}$ and
$v \models \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}$

Semantics for Conditional Propositional Logic

- Define when a valuation v satisfies a conditional proposition by

$$\begin{array}{l} v \models \mathbf{T} \\ v \not\models \mathbf{F} \\ v \models \mathit{if}\ A\ \mathit{then}\ P_t\ \mathit{else}\ P_f \text{ iff} \\ \qquad v(A) = \mathrm{true} \text{ and } v \models P_t \text{ or} \\ \qquad v(A) = \mathrm{false} \text{ and } v \models P_f \end{array}$$

- Example: let $v = \{A \mapsto \mathrm{true},\ B \mapsto \mathrm{true}, C \mapsto \mathrm{true}\}$

$v \models \mathit{if}\ C\ \mathit{then}\ \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{T}$ since
$v(C) = \mathrm{true}$ and
$v \models \mathit{if}\ B\ \mathit{then}\ \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}\ \mathit{else}\ \mathbf{F}$ since
$v(B) = \mathrm{true}$ and
$v \models \mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}$ since
$v(A) = \mathrm{true}$ and$v \models \mathbf{T}$

Translating Original Propositions into if_then_else

- Start with proposition P_0 with variables $v_1, \ldots v_n$
- $P[c/v]$ is the proposition resulting from replacing all occurrences of variable v with constant c
- Let $\overline{P}$ be the result of evaluating every subexpression of P containing no variables
- Let $P_1 = \mathit{if}\ \ v_1\ \ \mathit{then}\ \underline{\overline{P_0[\mathbf{T}/v_1]}}\ \mathit{else}\ \underline{\overline{P_0[\mathbf{F}/v_1]}}$
- Let $P_i = \mathit{if}\ \ v_i\ \ \mathit{then}\ \overline{P_{i-1}[\mathbf{T}/v_i]}\ \mathit{else}\ \overline{P_{i-1}[\mathbf{F}/v_i]}$
- P_n is logically equivalent to P, but only uses if_then_else_.
    - Valuation satisfies P if and only if it satisfies P_n
    - P_n depends on the order of variables $v_1, \ldots v_n$
    - P_n directly corresponds to a binary decision tree

Example:

$P = (A \wedge B) \vee (\neg C)$, variables $\{A,\ B,\ C\}$, $A < B < C$

$P_0 = (A \wedge B) \vee (\neg C)$

Example:

$P = (A \wedge B) \vee (\neg C)$, variables $\{A,\ B,\ C\}$, $A < B < C$

$$P_0 = (A \wedge B) \vee (\neg C)$$
$$P_1 = \mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ B) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ B) \vee (\neg C)$$

Example:

$P = (A \wedge B) \vee (\neg C)$, variables $\{A,\ B,\ C\}$, $A < B < C$

$$
\begin{aligned}
P_0 &= (A \wedge B) \vee (\neg C)\\
P_1 &= \textit{if}\ A\ \textit{then}\ (\mathbf{T}\ \wedge\ B) \vee (\neg C)\ \textit{else}\ (\mathbf{F}\ \wedge\ B) \vee (\neg C)\\
P_2' &= \textit{if}\ B\ \textit{then}\ (\textit{if}\ A\ \textit{then}\ (\mathbf{T}\ \wedge\ \mathbf{T}) \vee (\neg C)\ \textit{else}\ (\mathbf{F}\ \wedge\ \mathbf{T}) \vee (\neg C))\\
&\qquad \textit{else}\ (\textit{if}\ A\ \textit{then}\ (\mathbf{T}\ \wedge\ \mathbf{F}) \vee (\neg C)\ \textit{else}\ (\mathbf{F}\ \wedge\ \mathbf{F}) \vee (\neg C))
\end{aligned}
$$

Example:

$P = (A \wedge B) \vee (\neg C)$, variables $\{A,\ B,\ C\}$, $A < B < C$

$$\begin{aligned}
P_0 &= (A \wedge B) \vee (\neg C) \\
P_1 &= \mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ B) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ B) \vee (\neg C) \\
P_2' &= \mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ \mathbf{T}) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ \mathbf{T}) \vee (\neg C)) \\
&\qquad\quad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ \mathbf{F}) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ \mathbf{F}) \vee (\neg C)) \\
P_2 &= \mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg C)\ \mathit{else}\ \mathbf{F} \vee (\neg C)) \\
&\qquad\quad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg C)\ \mathit{else}\ \mathbf{F} \vee (\neg C))
\end{aligned}$$

Example:

$P = (A \wedge B) \vee (\neg C)$, variables $\{A,\ B,\ C\}$, $A < B < C$

$$
\begin{aligned}
P_0 &= (A \wedge B) \vee (\neg C) \\
P_1 &= \mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ B) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ B) \vee (\neg C) \\
P_2' &= \mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ \mathbf{T}) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ \mathbf{T}) \vee (\neg C)) \\
&\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ \mathbf{F}) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ \mathbf{F}) \vee (\neg C)) \\
P_2 &= \mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg C)\ \mathit{else}\ \mathbf{F} \vee (\neg C)) \\
&\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg C)\ \mathit{else}\ \mathbf{F} \vee (\neg C)) \\
P_3' &= \mathit{if}\ C\ \mathit{then}\ (\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg \mathbf{T})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{T})) \\
&\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg \mathbf{T})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{T}))) \\
&\qquad \mathit{else}\ (\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg \mathbf{F})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{F})) \\
&\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg \mathbf{F})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{F})))
\end{aligned}
$$

Example:

$P = (A \wedge B) \vee (\neg C)$, variables $\{A,\ B,\ C\}$, $A < B < C$

$$
\begin{aligned}
P_0 &= (A \wedge B) \vee (\neg C) \\
P_1 &= \mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ B) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ B) \vee (\neg C) \\
P_2' &= \mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ \mathbf{T}) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ \mathbf{T}) \vee (\neg C)) \\
&\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ (\mathbf{T}\ \wedge\ \mathbf{F}) \vee (\neg C)\ \mathit{else}\ (\mathbf{F}\ \wedge\ \mathbf{F}) \vee (\neg C)) \\
P_2 &= \mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg C)\ \mathit{else}\ \mathbf{F} \vee (\neg C)) \\
&\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg C)\ \mathit{else}\ \mathbf{F} \vee (\neg C)) \\
P_3' &= \mathit{if}\ C\ \mathit{then}\ (\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg \mathbf{T})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{T})) \\
&\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg \mathbf{T})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{T}))) \\
&\qquad \mathit{else}\ (\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T} \vee (\neg \mathbf{F})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{F})) \\
&\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F} \vee (\neg \mathbf{F})\ \mathit{else}\ \mathbf{F} \vee (\neg \mathbf{F}))) \\
P_3 &= \mathit{if}\ C\ \mathit{then}\ (\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}) \\
&\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F}\ \mathit{else}\ \mathbf{F})) \\
&\qquad \mathit{else}\ (\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{T}) \\
&\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{T}))
\end{aligned}
$$

Example, cont.

$$\begin{aligned} P_3 = \mathit{if}\ C\ \mathit{then}\ (&\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{F}) \\ &\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{F}\ \mathit{else}\ \mathbf{F})) \\ \mathit{else}\ (&\mathit{if}\ B\ \mathit{then}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{T}) \\ &\qquad\qquad \mathit{else}\ (\mathit{if}\ A\ \mathit{then}\ \mathbf{T}\ \mathit{else}\ \mathbf{T})) \end{aligned}$$

P_3 corresponds to second binary decision tree given earlier

- Any proposition in strict if_then_else_ form corresponds directly to a binary decision tree that accepts exactly the valuations that satisfy (model) the proposition.

Binary Decision Diagram

- Binary decision trees may contain (much) redundancy
- Binary Decision Diagram (BDD): Replace trees by (rooted) directed acyclic graphs
- Require all other conditions still hold
- Generalization of binary decision trees
- Allows for sharing of common subtrees.
- Accepts / rejects valuations as with binary decision trees.

Example

Reduced Ordered Binary Decision Diagrams

- Problem: given proposition may correspond to many different BDDs
- How to create a (compact) canonical BDD for a proposition such that two different propositions are logically equivalent if and only if they have the same (isomorphic) canonical BDD
- Start: order propositional variables $v_i < v_j$.
- Bryant showed you can obtain such a canonical BDD by requiring
    - Variables should appear in order on each path for root to leaf
    - No distinct duplicate (isomorphic) subtrees (including leaves)

Achieving Canonical Form

- Start with an Ordered BDD (all edges in correct order)
- Repeat following until none apply
- Remove duplicate leaves: Eliminate all but one leaf with a given label and redirect all edges to the eliminated leaves to the remaining one
- Remove duplicate nonterminals: If node n and m have the same variable label, their left edges point to the same node and their right edges point to the same node, remove one and redirect edges that pointed to it to the other
- Remove redundant tests: If both out edges of node n point to node m, eliminate n and redirect all edges coming into n to m
- Bryant gave procedure to do the above that terminates in linear time

Example