LOGIC-BASED TRUTH MAINTENANCE SYSTEMS

EECS 344

Winter, 2008

Overview

- Limitations of the JTMS
- LTMS basics
- Logical Specification of LTMS
- Boolean Constraint Propagation
- Interface to inference engine
- Example: Constraint solving

Logical import of JTMS clauses

- ***Definite clauses***

$$x_1 \wedge \ldots \wedge x_n \Rightarrow c$$

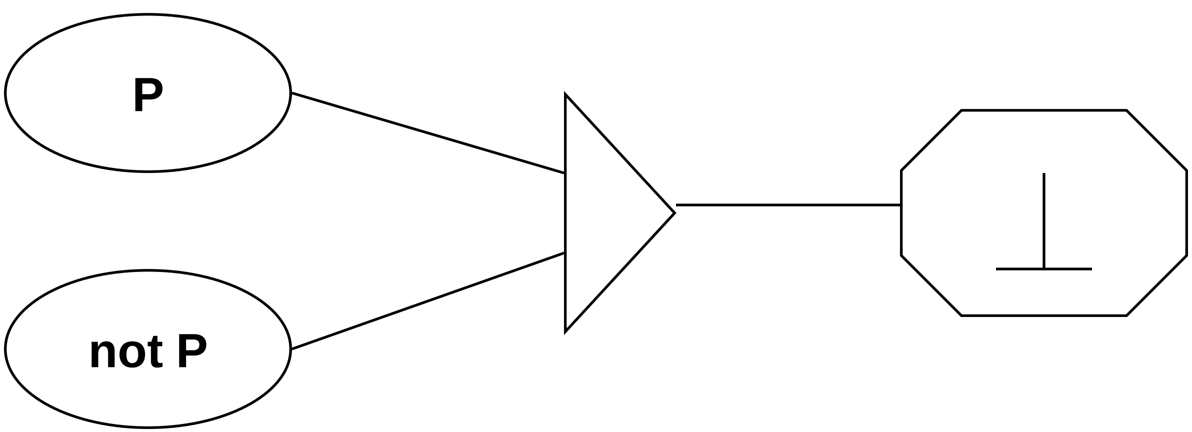
- **No negation**
- **Cannot directly say**

$$x \Rightarrow \neg y$$

- **Must use encoding tricks to implement more expressive logic**

Encoding negation in JTMS

- For each propositional node P, add extra node for its negation.
- Install a justification for a contradiction for P and its negation.



Encoding Arbitrary Clauses

- **Suppose we want to encode**

$$A \vee B \vee C$$

- **Could translate into a set of definite clauses**

$$A \wedge \neg A \Rightarrow \bot \quad \neg B \wedge \neg C \Rightarrow A$$

$$B \wedge \neg B \Rightarrow \bot \quad \neg A \wedge \neg C \Rightarrow B$$

$$C \wedge \neg C \Rightarrow \bot \quad \neg A \wedge \neg B \Rightarrow C$$

All clauses require expansion

- **Consider `(implies P Q)`**

$$P \Rightarrow Q$$

$$\neg Q \Rightarrow \neg P$$

- **Especially important in backtracking, if information can be derived in different orders**

$$A_1 \wedge \ldots \wedge A_{i-1} \wedge A_{i+1} \wedge \ldots \wedge A_n \Rightarrow \neg A_i$$

Solution: Use a more powerful TMS

- **Nodes have three possible labels:**
  - **:TRUE**
  - **:FALSE**
  - **:UNKNOWN**
- **Justifications are disjunctive clauses:**

$$\neg p \vee \neg q \vee r$$

- **Each term in a clause has a *sign,* e.g., whether or not it is negated**

Other LTMS modifications

- **Assumptions work as before**
- **Premises work as before (i.e., they are nodes justified by empty clauses)**
- **No contradiction nodes are necessary**
- **Contradiction detection is handled by clauses being *violated***

$$A \vee \neg B$$

$$\neg A$$

$$B$$

Logical Specification of LTMS

- **Given**
  - **a set of clauses *C***
  - **a set of assumptions *A***
- **For any proposition P, label it**
  - **`:TRUE` if it is derivable**
  - **`:FALSE` if its negation is derivable**
  - **`:UNKNOWN` otherwise**
- **If *C* & *A* are unsatisfiable, complain**
- **Produce explanations for every labelled node, even when *C* & *A* unsatisfiable.**

Boolean Constraint Propagation

- Best algorithm for implementing an LTMS
- Sound
- Efficient
- Incomplete (but see Chapter 13!)

Basic Idea

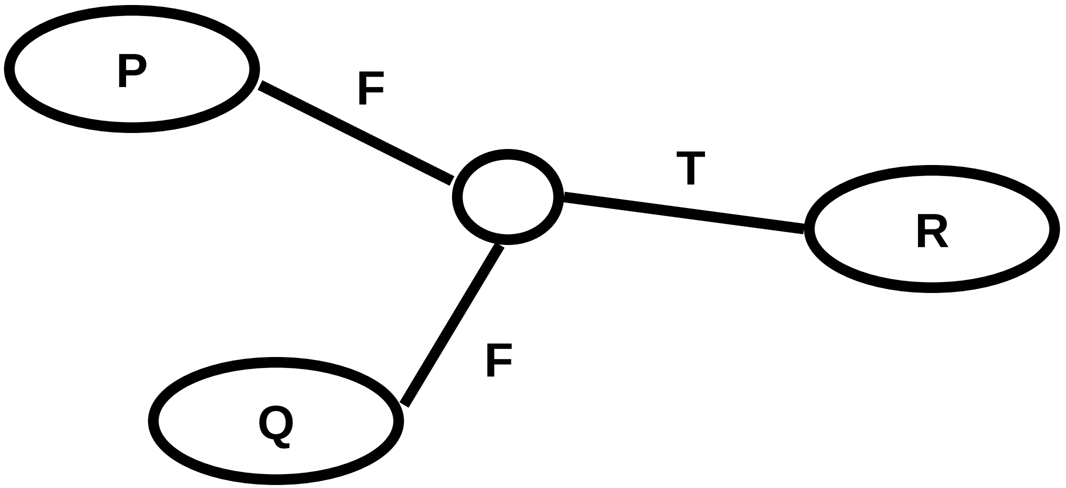
- A clause is either
  - *Satisfied*: Some node's sign matches its label
  - *Violated*: Every node's sign is opposite that of its label
  - *Unit Open*: One node is unknown, remainder have signs opposite their labels.
  - *Non-Unit Open*: Multiple unknown nodes, clause unsatisfied.
- Observation #1: A unit open clause can be satisfied by labeling it with its sign.
- Observation #2: A violated clause indicates a contradiction.
- Observation #3: No other cases allow inference.

Example

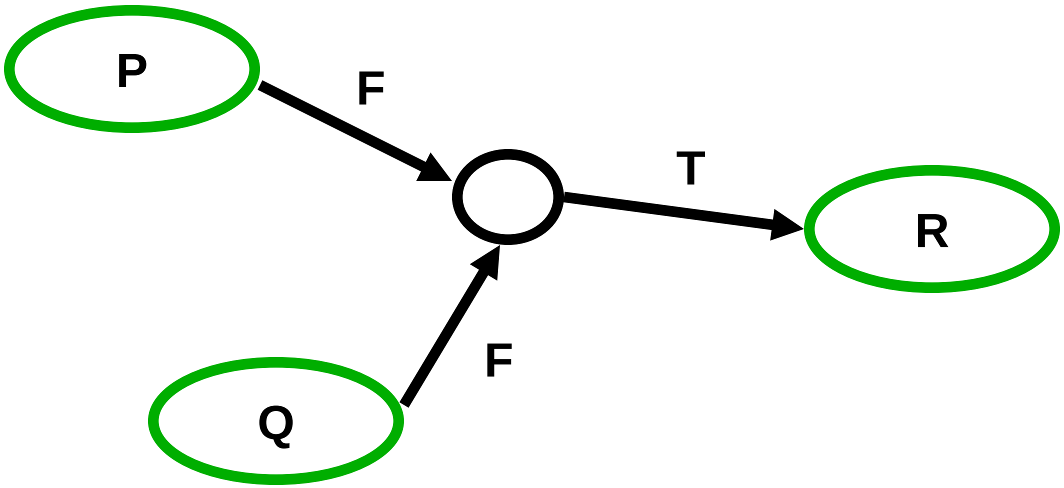
$$\neg p \vee \neg q \vee r$$

- **P false, satisfied.**
- **P true, Q true, R false, violated.**
- **P true, Q true, R unknown: Unit open. Can derive R as true**
- **P unknown, Q true, R false: Unit open. Can derive P as false.**

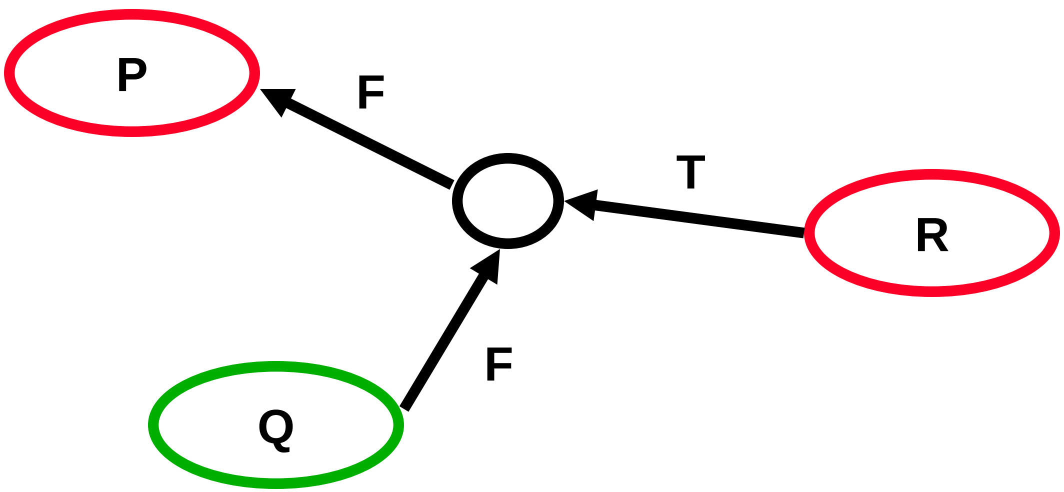
Graphical notation for LTMS



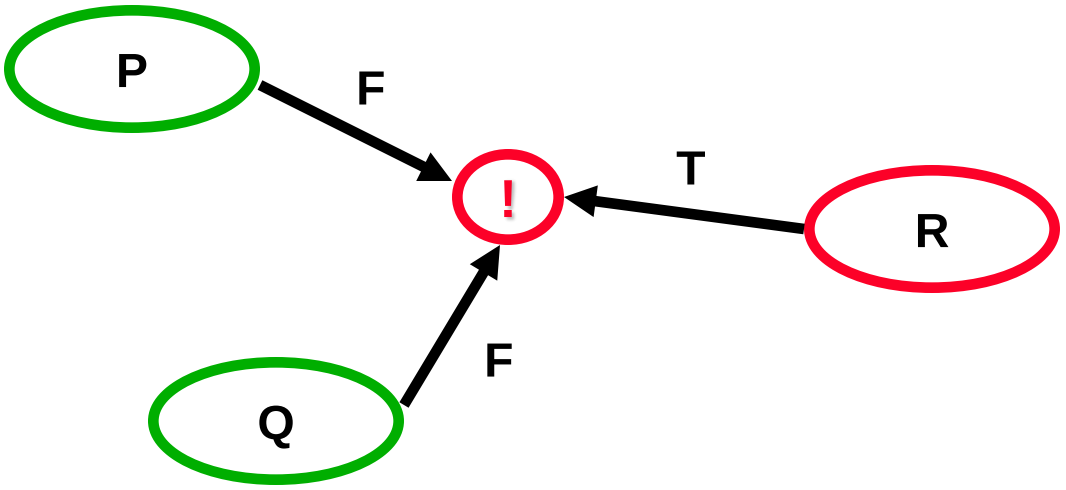
Clauses dynamically simulate definite clauses



Clauses are multidirectional



Clauses provide contradiction detection



Limitations of BCP

- **Literal incompleteness**

$$x \vee y$$
$$x \vee \neg y$$

- **Refutation incompleteness**

$$x \vee y \qquad \neg x \vee y$$
$$x \vee \neg y \qquad \neg x \vee \neg y$$

- **No formal characterization of when it loses**

Inference Engine Interface

- Interpretation of labels
- How to specify clauses
- Adding data
- Queries
- Rules
- Contradiction Handling

Using more complex labels

- **No longer have `(:NOT P)` in the database**
- **In querying `(:NOT P)`, return opposite of label of `P`**
- **When asserting/assuming `(:NOT P)`, `P` becomes a premise/assumption with label `:FALSE`**

Automatic translation into clauses

- **External system uses standard propositional logic, with usual set of connectives (plus `taxonomy`)**
- **LTMS code translates into appropriate set of clauses (see `normalize` in `ltms.lisp`)**
- **Original form used as informant for explanations**
- **Only time a statement with connectives is entered into the database is if it is assumed.**

Warning: A common bug

- The set of connectives is
  - `:NOT, :AND, :OR, :IMPLIES, :IFF, :TAXONOMY`
- These aren't connectives
  - `NOT, AND, OR, IMPLIES, IFF, TAXONOMY`
  - `=>, ~`

Adding Data

- **`assert!, assume!, retract!, rassert!` as before**
- **`contradiction` takes a list of nodes and creates a clause that is violated, given their current labels.**
- **`assuming` is a macro that provides an environment with temporary assumptions**

Queries

- Assertion-level queries (e.g., `fetch`, `referent`) equivalent.
- Queries about beliefs now reflect new labels (i.e., `true?`, `false?`, `known?`, `unknown?`)
- Explanation-exploring procedures similar to before (e.g., `why?`, `assumptions-of`, `consequences`, `explore`)

Rules

- Trigger conditions now reflect belief states (e.g., `:TRUE`, `:FALSE`, but not `:UNKNOWN`)
- Otherwise identical to earlier systems

```
(rule ((:true (human ?x) :var ?h))
 (rassert! (:implies ?h
            (:and (mortal ?x)
                  (:not
                   (infallible ?x))))
           :limited-creatures))
```

Contradiction Handling

- Orthogonal issue to type of TMS
- Before: lambda-bind single contradiction handler.
- Not good enough!

Example

- **Consider the following choice sets:**
  - **{A1, A2, A3}**
  - **{B1, B2, B3}**
  - **{C1, C2, C3}**
- **Suppose each set has its own contradiction handler (Ha, Hb, Hc)**
- **Suppose we are exploring {A2, B2, C2}**
- **Suppose we find a contradiction whose underlying assumptions is {A2, B2}.**
- **We’re in trouble -- why should Hc know what to do here?**

When does rebinding work?

1. **All assumption-manipulating operations are identified, and each provided with an appropriate contradiction handler.**

2. **Assumption-manipulating operations must proceed depth-first.**

3. ***Relative Closure:*** **Every consequence that holds for the current set of assumptions that might lead to a contradiction must be computed before making more assumptions.**

Relative Closure often unrealistic

- Information can arrive unexpectedly
- The set of consequences can be infinite
- Processing can be distributed

- Often works for toy problems
- But it should be abandoned very quickly!

Solution: Stack-based contradiction handling

- Organize assumption-manipulating operations in depth-first fashion
- Each operation pushes a contradiction handler when it begins, and pops it when it is finished.
- When a contradiction occurs, check each handler in turn to see if it is relevant.
- Implements chronological backtracking within subset of relevant choices.

Assume a particular failure

Assume that you know how the parts can fail

Assume that you know how the parts work

Assume parts you know about are the only relevant ones

Assume a repairable part is the source of the problem

Example: Simple constraint satisfaction problem

- **Kind of problem often found in “logic books” in newsstands**
- **Formally, set of variables whose values range over a finite domain (mathematical perspective).**
- **Formally, a set of attribute statements about a collection of objects (logical perspective).**

Example: Remember the Marx Brothers?

- Groucho, Chico, Harpo, and …?
- One liked to expound, another played the piano, another liked animals…
- Which one was which?

Constraints

- The pianist, harpist, and talker are distinct brothers.
- The brother who is fond of money is distinct from the one who is fond of gambling, who is also distinct from the one who is fond of animals.
- The one who likes to talk doesn't like gambling.
- The one who likes animals plays the harp.

More constraints

- **Groucho hates animals.**
- **Harpo is always silent.**
- **Chico plays the piano.**

Homework

- **Problem 7(b), page 343**
- **Test problems:**

```
  SEND         DONALD     FIFTY     BASE
+ MORE       + GERALD    +STATES    +BALL
----------- --------    -------     -----
 MONEY        ROBERT     AMERICA    GAMES
```

- **Hints:**
  - **Think hard about representation first!**
  - **Squeeze as much information out as possible when making each assumption**
- **Optional: For background, see http://www.geocities.com/Athens/Agora/2160/primer.html**