Non Monotonic Logics

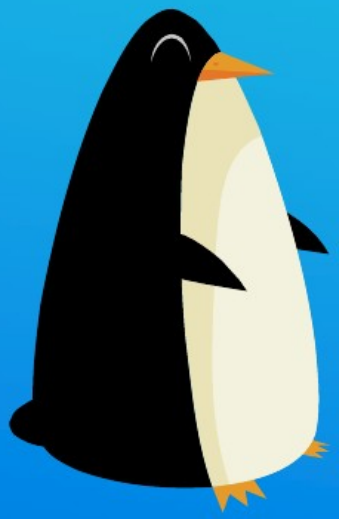
Prof. Mauro Gaspari
Dipartimento di Informatica Scienze e Ingegneria (DISI)
mauro.gaspari@unibo.it

Problems with FOL

- The size of FOL KB necessary to describe real situations is almost always too large.
- Often available knowledge is incomplete and FOL has not specific mechanisms to deal with incomplete knowledge. For example FOL is too rigid to deal with new (conflicting) knowledge.
- To model commonsense reasoning, it is necessary to be able to jump to plausible conclusions from the given knowledge making assumptions.
- The choice of assumptions is not blind: most of the knowledge on the world is given by means of general rules which specify typical properties of objects. For instance, "birds fly" means: birds typically fly, but there can be exceptions such as penguins, ostriches, ...

Example

$$\forall x.Bird(x) \rightarrow Fly(x)$$

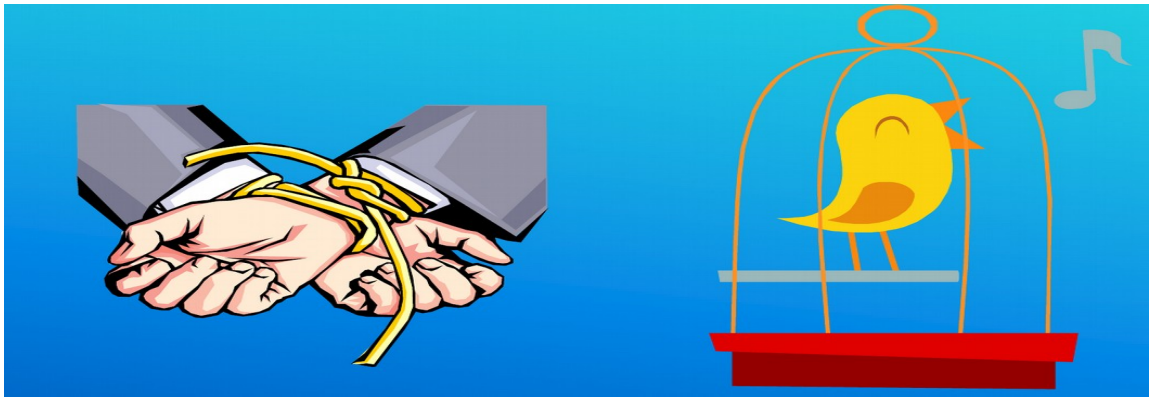


FOL solution

- We can try to capture this semantics adding more conditions:

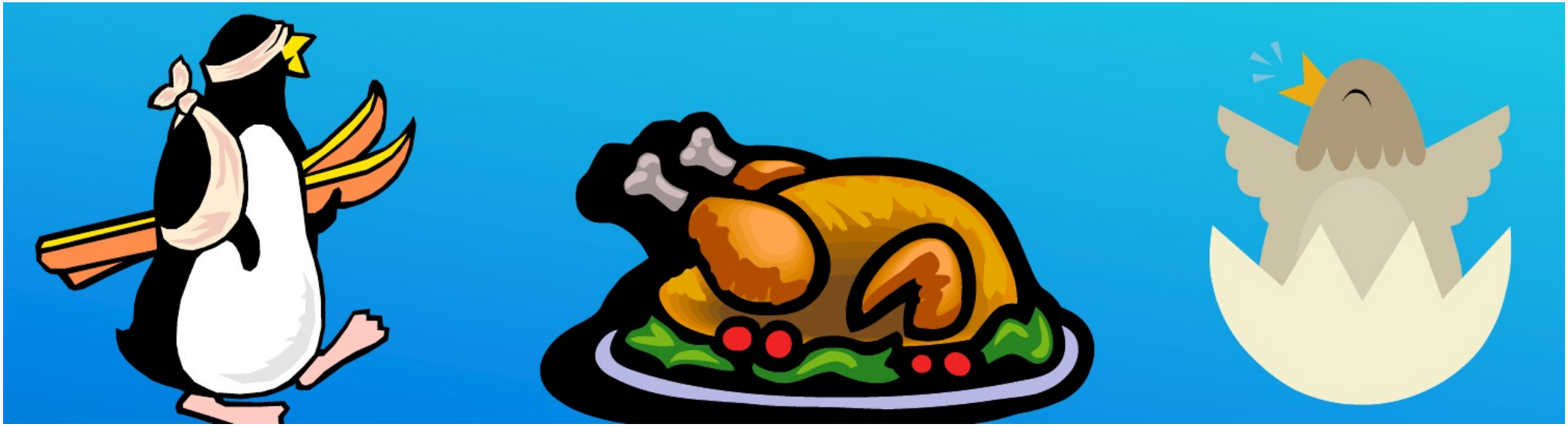
$$\forall x\ Bird(x) \wedge \neg Penguin(x) \wedge \neg Ostrich(x) \wedge \neg Peacock(x) \wedge \neg Chicken(x) \rightarrow Fly(x)$$

- But this is not enough: indeed in general situations additional exceptions should be considered.



$$\forall x\ Bird(x) \wedge \neg Penguin(x) \wedge \neg Ostrich(x) \wedge \neg Peacock(x) \wedge \neg Chicken(x) \wedge \neg Tied(x) \wedge \neg Caged(x) \wedge \neg Cemented(x) \rightarrow Fly(x)$$

And More



- ∀x Bird(x) ∧¬Penguin(x)∧¬Ostrich(x)
  ∧¬Peacock(x)∧¬Chicken(x)∧¬Tied(x)
  ∧¬Caged(x)∧¬Cemented(x)∧¬Injured(x)
  ∧¬Dead(x)&-Baby(x)→Fly(x)

Observation

- It is always possible to add more detail to the environment but the Frame Problem arises.
- In general it is not possible to know which are the right exceptions to add. It is possible only if the possible queries are known.

Nonmonotonic Reasoning

- **Nonmonotonic reasoning** deals with the problem of deriving plausible conclusions, but not infallible, from a KB.
- Since the conclusions are not certain, it must be possible to retract some of them if new information shows that they are wrong.
- Classical logic is inadequate since it is monotonic: if a formula B is derivable from a set of formulas S , then B is also derivable from any superset of S :

  $S \vdash B$ implies $S \cup \{A\} \vdash B$, for any formula A.

- Nonmonotonic logics have been devised with modified notions of truth and entailment in order to capture such behavior.

Example

- Let the KB contains:

  Typically birds fly.

  Penguins do not fly.

  Tweety is a bird.

- We can infer that Tweety flies.
- However if the following information is added to the KB:

  Tweety is a penguin

- the previous conclusion must be retracted and, instead, the new conclusion that Tweety does not fly will hold.

Nonmonotonic Reasoning

- The statement "typically A" can be read as: "in the absence of information to the contrary, assume A".
- The problem is to define the precise meaning of "in the absence of information to the contrary ".
- The meaning could be: "there is nothing in KB that is inconsistent with assumption A".
- However, other interpretations are possible
- Different interpretations give rise to different non-monotonic logics

Nonmonotonic logics

- Non-Monotonic logics have been proposed from the beginning of the 80's, here are historically the most important proposals:
  - **Closed World Assumption**.
  - **Circumscription, by McCarthy, '80**
  - Non-monotonic logic, by McDermott and Doyle, '80
  - **Default Logic, by Reiter, '80**
  - Autoepistemic logic, Moore '84

Closed World Assumption

- A basic understanding of the logic used in databases, is that only positive information is represented explicitly and negative information is not represented explicitly.
- A more sophisticated KR system might allow the user to specify rules for which to apply the Closed World Assumption (CWA). This logic can be also applied to Kbs. If a positive fact is not present in a KB, it is assumed that its negation holds.
- This is called Closed World Assumption: the only true facts are the provable ones, this is the principle of negation as (finite) failure.
- If KB $\nvdash$ A then ~~DB~~ $\vdash_{CWA} \neg A$
- Note that, this inference is not valid in classical logic.

Example

- Suppose a KB contains facts of the form "practice(person, sport)"':

  practice(anne, tennis)

  practice(joe, tennis)

  ractice(anne, sky)

- Then we have:

  ~~DB~~ $\vdash_{CWA}$ ¬practice(joe, sky)

- Trivially CWA is non-monotonic, since adding a fact may lead to withdraw the negative conclusion:

  - ~~DB~~ ∪ {practice(joe, sky)} $\nvdash_{CWA}$ ¬practice(joe, sky)

Circumscription

- **Circumscription** can be seen as a more powerful and precise version of the closed world assumption.
- The idea is to specify particular predicates that are "as false as possible": false for every object expect those for which they are known to be true.
- For example, suppose we want to assert the default rule that birds fly. This can be done introducing a predicate $\mathsf{Abnormal}_1(x)$:

  $\forall x\ \mathsf{Bird}(x) \wedge \neg \mathsf{Abnormal}_1(x) \rightarrow \mathsf{Flies}(x)$

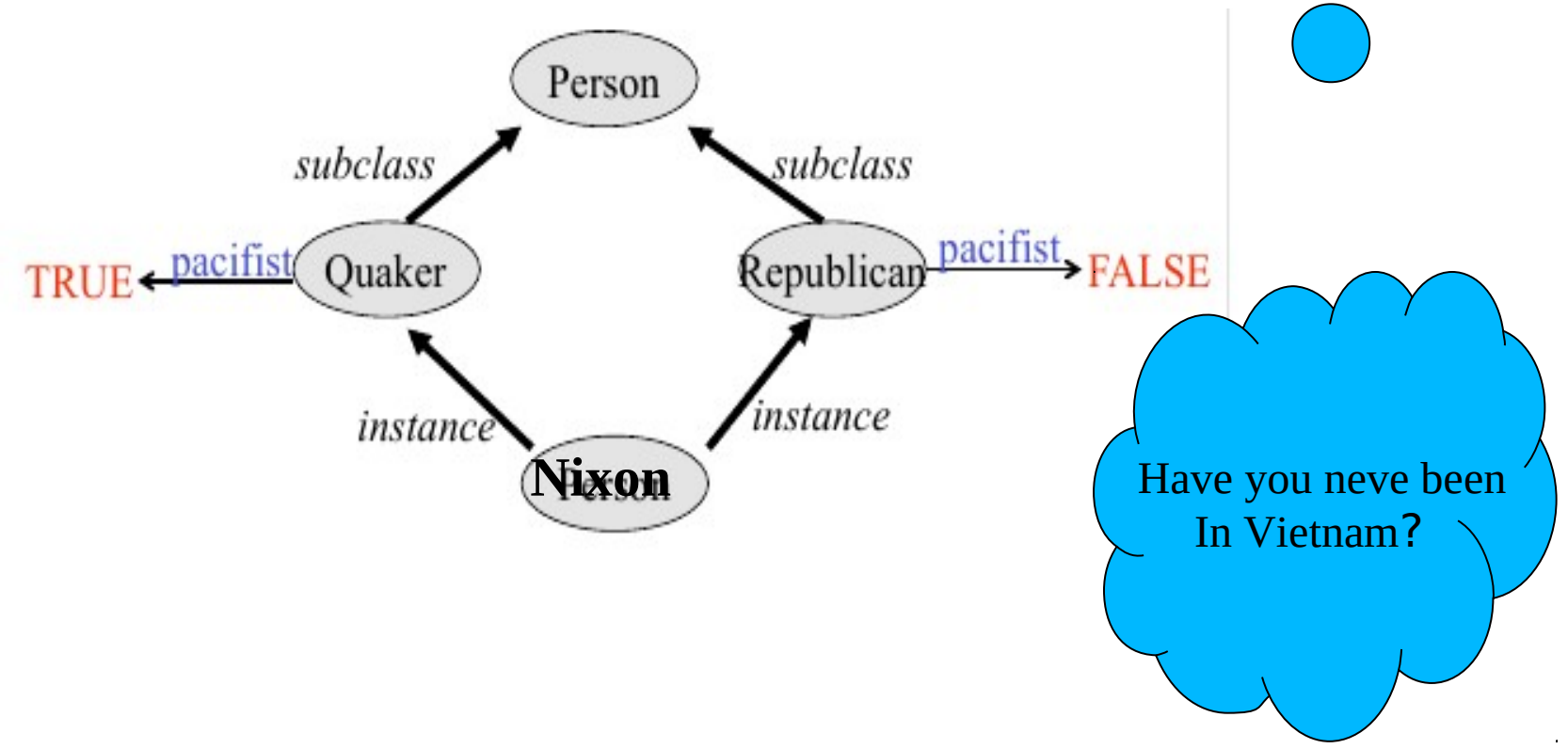
- If we specify that $\mathsf{Abnormal}_1$ is to be circumscribed a circumscriptive engine assumes $\neg\mathsf{Abnormal}_1(x)$ unless $\mathsf{Abnormal}_1(x)$ is known to be true.
- We can infer $\mathsf{Flies}(\mathsf{Tweety})$ from $\mathsf{bird}(\mathsf{Tweety})$, but the conclusion no longer holds if $\mathsf{Abnormal}_1(\mathsf{Tweety})$ is asserted.

Model Preference Logics

- Circumscription can be viewed as an example of **model preference logic**.
- In these logics a sentence is entailed (as a default) if it is true in all preferred models of the KB (in classical FOL it should be true in all models).
- A model is preferred to another if if has fever abnormal objects.

Example: Nixon Diamond

- This approach can be exploited to model multiple inheritance in the context of semantic networks.
- The Nixon diamomd: Is Nixon a Pacifist?



Example

- We can encode this knowledge in the following KB:

  $\text{Republican}(\text{Nixon}) \wedge \text{Quaker}(\text{Nixon})$
  $\text{Republican}(x) \wedge \neg \text{Abnormal}_2(x) \rightarrow \neg \text{Pacifist}(x)$
  $\text{Quaker}(x) \wedge \neg \text{Abnormal}_3(x) \rightarrow \text{Pacifist}(x)$

- If we circumscribe Abnormal_2 and Abnormal_3 there are two preferred models: 1. $\text{Abnormal}_2(\text{Nixon})$, $\neg \text{Pacifist}(\text{Nixon})$
  2. $\text{Abnormal}_3(\text{Nixon})$, $\text{Pacifist}(\text{Nixon})$

- Thus the circumscriptive engine remains properly agnostic.

- If we want to assert thet religion beliefs take precedence over political beliefs we can use a formalism called **prioritized circumpcription**.

Preferential Models

- Define a pair <L,$\sqsubset$> where L is a non monotonic logic (based on FOL) and $\sqsubset$ is a partial order on the models.
- $M_1 \sqsubset M_2$ reads as M_2 is preferred over M_1.
- If M is a model of KB, M is a preferred model if no other model M' is a model of KB and $M \sqsubset M'$.
- We say that $KB \mid\sim_{\sqsubset} P$ iff P is true for all preferred models of KB.

Default Logic

- Default logic extends FOL inference rules by non-standard **default rules**. These rules allows one to express default properties.
- Example:

$$\frac{bird(x) : flies(x)}{flies(x)}$$

- This rule can be interpreted as: "if x is a bird and we can consistently assume that x flies (flies(x) is consistent with the KB) then we can infer that x flies".

Default Rules

- In general a Default Rule has the form:

$$\frac{\alpha(x) \ : \beta(x)}{\gamma(x)}$$

- that can be interpreted as: "'if α(x) holds and β(x) can be consistently assumed then we can conclude γ(x)".
  - α(x): the prerequisite
  - β(x): the justification
  - γ(x): the consequent
- It is possible to specify more than one justification: in any one of them can be proven false, then the conclusion cannot be droawn.

Default theory

- A default theory is a pair < D,W>, where D is a set of default rules and W is a set of first-order formulas.
- Example 1: let <D,W> be:

$$D = \{\frac{bird(x) \ : fly(x)}{fly(x)}\}$$

$$W = \{bird(tweety), \forall x(penguin(x) \rightarrow bird(x)), \forall x(penguin(x) \rightarrow \neg fly(x))\}$$

Extension

- Intuitively, in a default theory <D,W>: W represents the stable (but incomplete) knowledge of the world
- D rules for extending the knowledge W by plausible (but defeasible) conclusions.
- Notion of **extension** of a default theory: the theory (= deductively closed set of logical formulas) obtained by extending W by the rules in D .

Example 2

- Let <D,W> be as in the previous example 1.
- Since bird(tweety) is true, and it is consistent to assume fly(tweety), then fly(tweety) is true in the (unique) extension of <D,W>.
- Consider now the the default theory <D,W′>, where W′= W ∪ {penguin(tweety)}
- then the assumption fly(tweety) is no longer consistent, and the application of the default rule is blocked.

Example 3

- Let <D,W> be as follows:

$$D = \{d_1 = \frac{Rep(x) \ : \neg Pac(x)}{\neg Pac(x)}, d_2 = \frac{Quack(x) \ : Pac(x)}{Pac(x)}, \}$$

$$W = \{Rep(Nixon), Quack(Nixon)\}$$

- For both default rules d_i, the prerequisite is derivable from W. What can be concluded from <D,W>?
- If we apply d_1, we conclude ¬Pac(Nixon); therefore Pac(Nixon) cannot be assumed consistently, so that d_2 is blocked.
- If we apply d2, we conclude Pac(Nixon); therefore ¬Pac(Nixon) cannot be assumed consistently, so that d1 is blocked.

Observations

- There are two extensions: one containing ¬Pac(Nixon) and the other containing Pac(Nixon).
- An extension (to be defined next) represents the set of plausible conclusions.
- In summary a default-theory may have zero, one, or many extensions.
- How to deal with this issue?

Inference

- Since a default theory $\Delta =< W, D >$ may have multiple extensions (including none), how to define a notion of inference?
- There are two natural notions:
  - **Credulous inference:** $\Delta \vdash_c A$ if there exists an extension E of Δ such that $A \in E$.
  - **Skeptical inference**: $\Delta \vdash_s A$ if for all extensions E of Δ, we have $A \in E$.
- Note that, since a default theory may have no extensions $\Delta \vdash_s A$ does not imply $\Delta \vdash_c A$.

Truth Maintenance Systems

- We have argued that many of the inferences drawn from a KRS will have only default status rather than being absolutely certain.
- **Belief revision** is sometimes necessary: when some of the inferred facts turn out to wrong.
- A **truth maintenance system** (**TMS**) is a program that keeps track of dependencies between sentences so that retraction (and some other operations) will be more efficient.

A Simple TMS

- The simplest approach to truth maintenance is to use **chronological backtracking**.
- In this approach, we keep track of the order in which sentences are added to the knowledge base by numbering them from P_1 to P_n. When the call RETRACT(KB,P_i) is made, the system reverts to the state just before P_i was added. Removing P_i and the sentence derived from it.
- If desired, the sentences P_{i+1} through P_n can then be added again. This is simple, and it guarantees that the knowledge base will be consistent, but it means that retraction is O(n), where n is the size of the knowledge base.
- We would prefer a more efficient approach that does not require us to duplicate all the work for P_{i+1} to P_n.

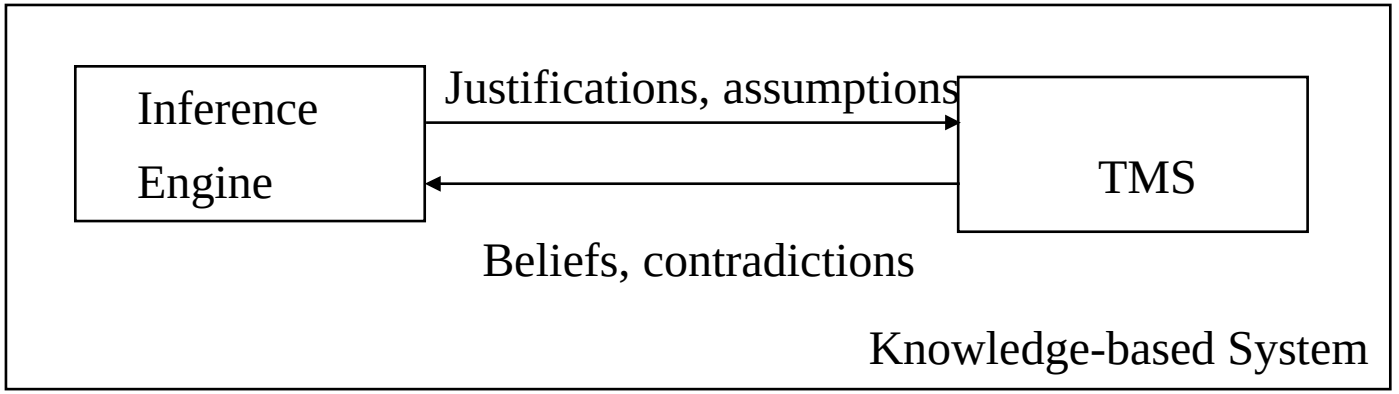
TMS Functionalities

- A TMS actually performs four important jobs:

1. It enables dependency-directed backtracking, to avoid the inefficiency of chronological backtracking.
2. It provides explanations of propositions. A proof is one kind of explanation—if we ask, "Explain why you believe P is true?" then a proof of P is a good explanation. If a proof is not possible, then a good explanation is one that involves assumptions.
3. It support default reasoning.
4. It helps in dealing with inconsistencies. If adding P to the KB results in a logical contradiction, a TMS can help pinpoint an explanation of what the contradiction is.

TMS

- A Truth Maintenance System (TMS) is module of a KRS that can be integrated with an engine providing the described functionalities.



TMS Types

- There are several types of TMSs:
- **JTMS (*justification-based* TMS)**: is the simplest system. In a JTMS, each sentence in the knowledge base is annotated with a justification that identifies the sentences from which it was inferred, if any. For example, if Q is inferred by Modus Ponens from P, then the set of sentences {P, P → Q} could serve as a justification of the sentence Q.
- **ATMS (*assumption-based* TMS)**: it is more powerful and maintains information at each node within a single dependency network for all the alternative sets of assumptions that we may hold, also called worlds or environments.
- **LTMS (*logic-based* TMS)**: Nodes have three possible labels (:TRUE, :FALSE, :UNKNOWN) and Justifications are disjunctive clauses.

Enforcing logical relations among beliefs

- Most of AI problem solvers are based on search
- Changing assumptions requires updating consequences of beliefs
- Rederivation can be expensive
  - Large, complex calculations (e.g., computational fluid dynamics)
  - Physical experiments

Recover from Inconsistencies

- Data can be wrong
  - The patient's temperature is 986 degrees.
- Constraints can be impossible
  - Our new computer should
    - »Run off batteries for 8 hours
    - »Fit in a shirt pocket
    - »Run faster than a Cray MP-X

Explanations

- Providing answers is not enough
  - Cut the patient's heart out
  - That design won't work
- Explanations are needed
  - Radical bypass surgery is required because...
  - No material will stand the projected stresses.

Guide Backtracking

- (Dependency-directed backtracking vs Chronological backtracking).
- Avoid rediscovering contradictions
- Avoid throwing away useful results

Example

Choose in sequence:

- A or B
- C or D
- E or F

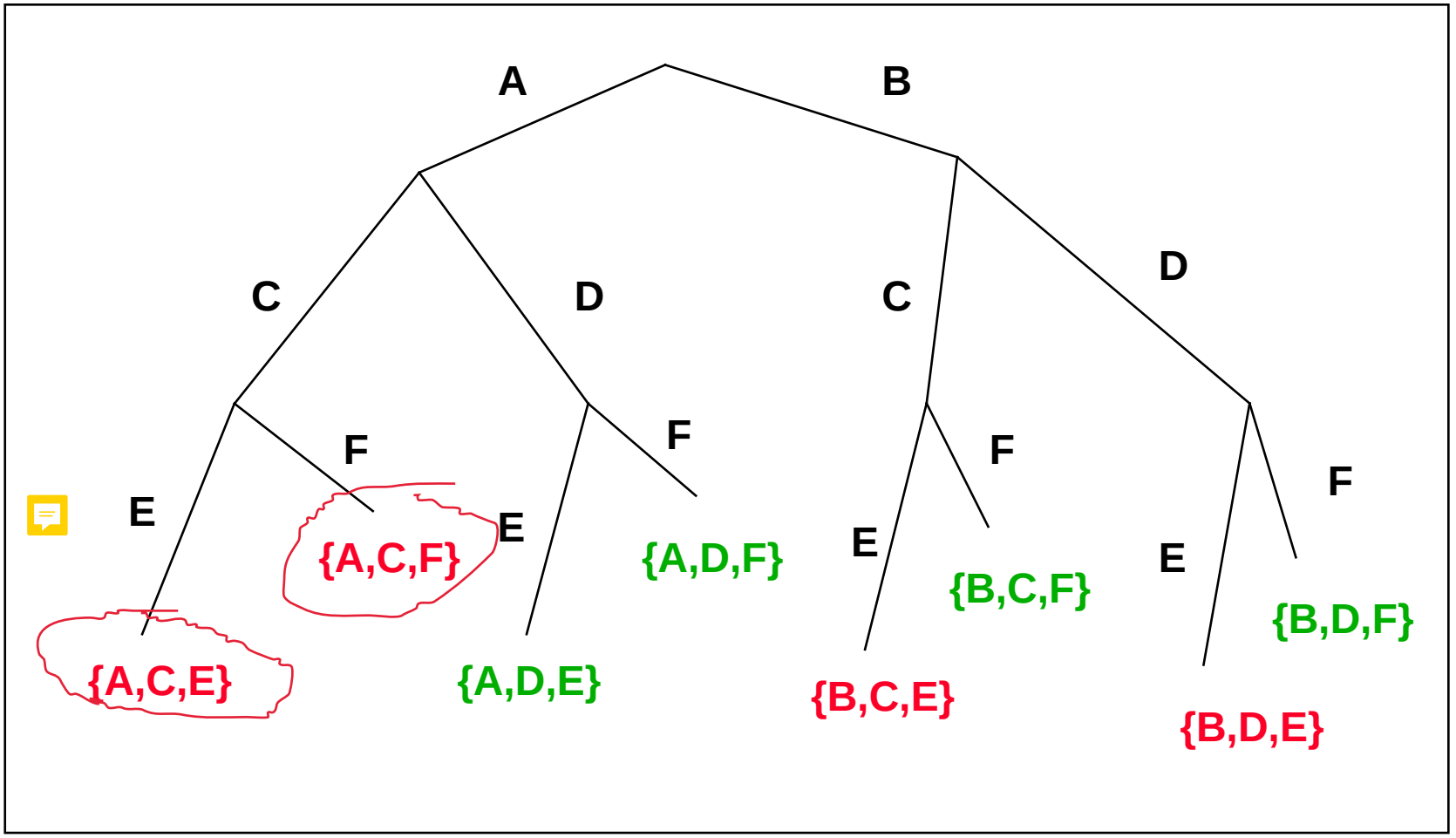
Given: A and C cannot hold together

Given: B and E cannot hold together

Assume we want all consistent solutions

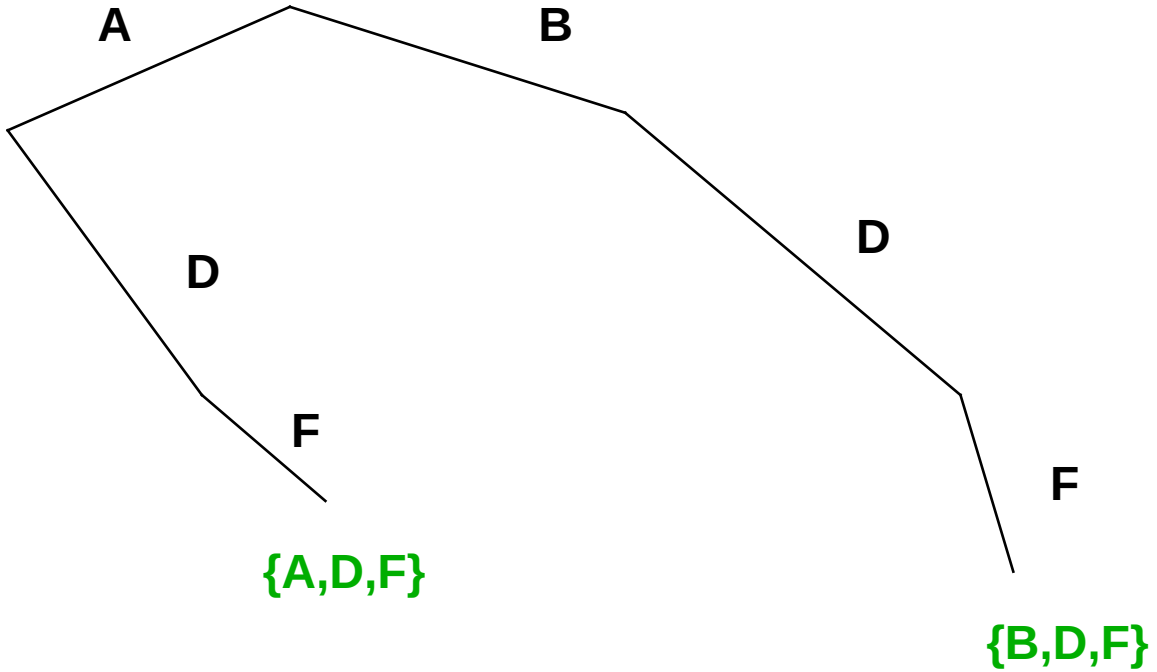
Assume that we cannot test until every choice has been made

Search Space



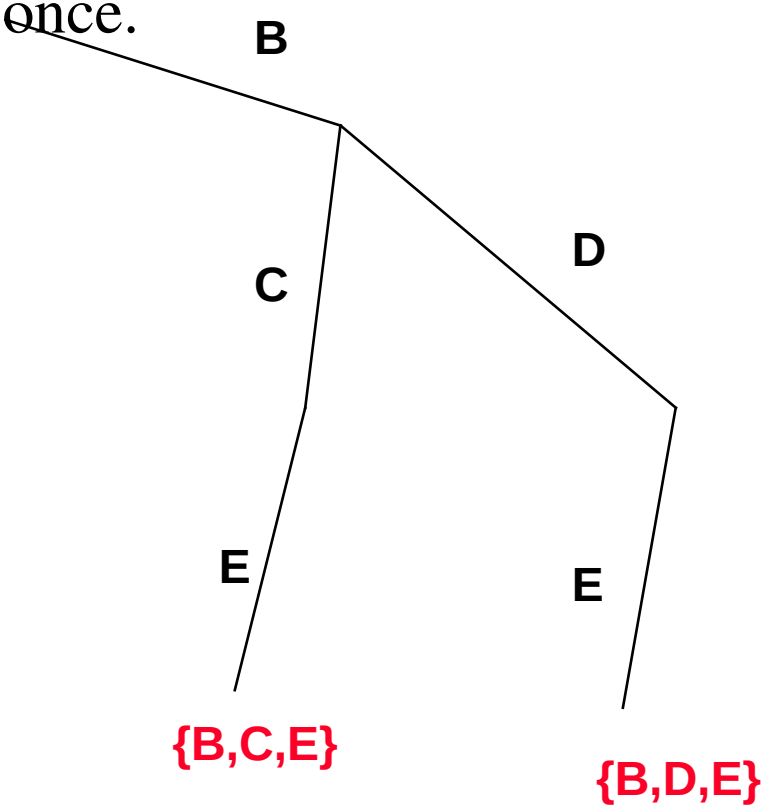
Chronological Backtracking

- Often wastes computation: Suppose that D and F together cause lots of work. Popping context loses this work



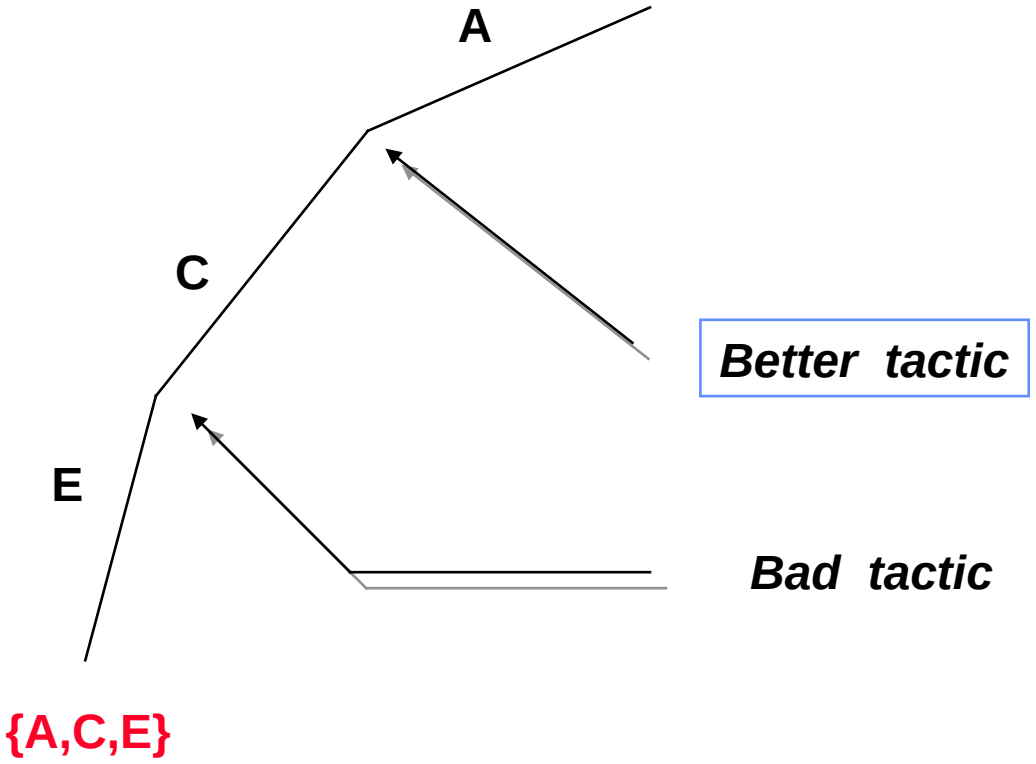
Chronological Backtracking

Rediscovers Contradictions: we can avoid to try B and E together more than once.



Guide backtracking

- Dependencies can guide backtracking:



JTMS Idea

- *Justifications* express relationships between beliefs
  - Justifications for a belief provide explanations, ability to pinpoint culprits
- Belief in an assertion expressed via its *label*
  - P being in database no longer is the same as believing P
  - Assertions and justifications serve as cache
  - Rules/other computations need only be executed once
- Justifications can be used to record inconsistencies
  - *Dependency-directed backtracking*
- Defaults can be represented via explicit assumptions

JTMS nodes

- Each belief is represented by a TMS node
- TMS nodes are associated 1:1 with assertions
- The label of a node represents the belief status of the corresponding problem solver fact.
- The relationships between beliefs are expressed by the justifications it participates in.

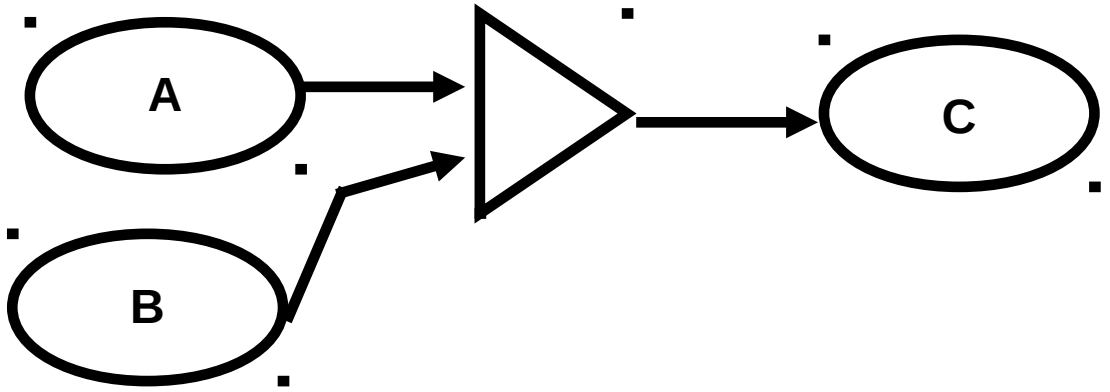
JTMS Labels

- Every assertion is either IN or OUT
  - IN = “believed”
  - OUT = “not believed”
- Warning: IN does not mean TRUE

	P in	P out
(not P) in	Contradiction	(not P) true
(not P) out	P true	Don’t know

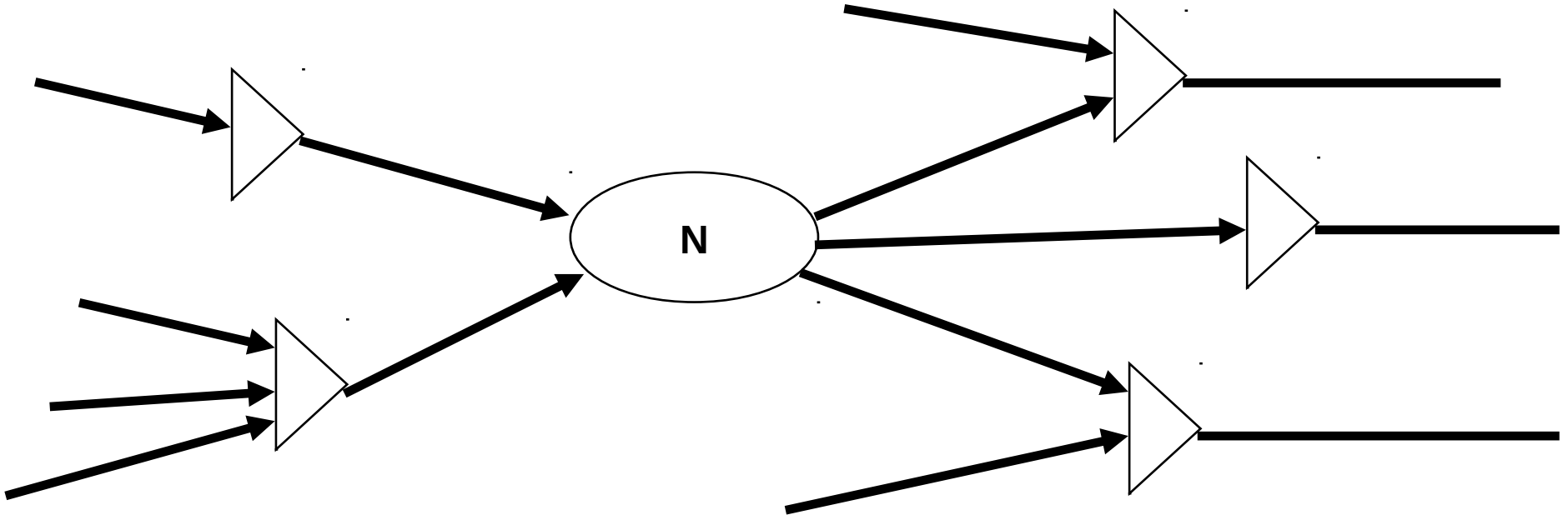
JTMS Justifications

- Must be Horn clauses
- Definitions
  - ***Consequent*** is the node whose belief is supported by a the justification
  - ***Antecedents*** are the beliefs which, when IN, support the consequent
  - ***Informant*** records information from external systems



Dependency Networks

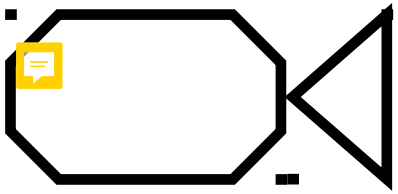
- Each node has:
  - *Justifications:* the justifications which have it as the consequent
  - *Consequences:* justifications which use it as an antecedent
  - *Support:* a single justification taken as the reason for it being IN, if any.



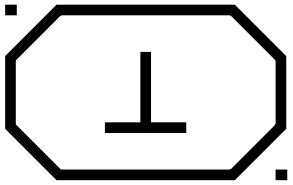
Special states of JTMS nodes

- *Assumptions* are IN if enabled.
- *Premises* are always IN.
- *Contradictions* should never hold.

Assumption



Premise



Contradiction

Enforcing constraints between beliefs

- A node is IN when either:
  1. It is an enabled assumption or premise
  2. There exists a justification for it whose antecedents are all IN
- Assumptions underlying a belief can be found by backchaining through supporting justifications
- JTMS operations must preserve *well-founded support*.

A TMS operates incrementally

- At any time, the inference engine can add
  - new justifications
  - delcare a statement to be a premise or contradiction (permanently)

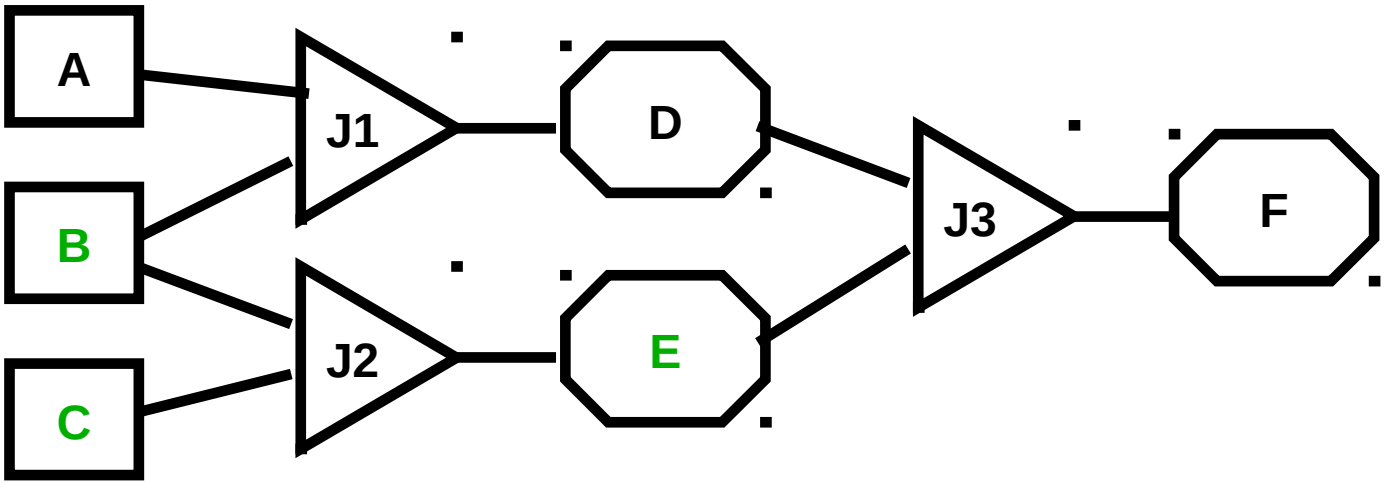
  –Assume a statement
- In all cases,

  1. Set the directly affected node, if any.
  2. Propagate the consequences (`Propagate-inness`)

Example

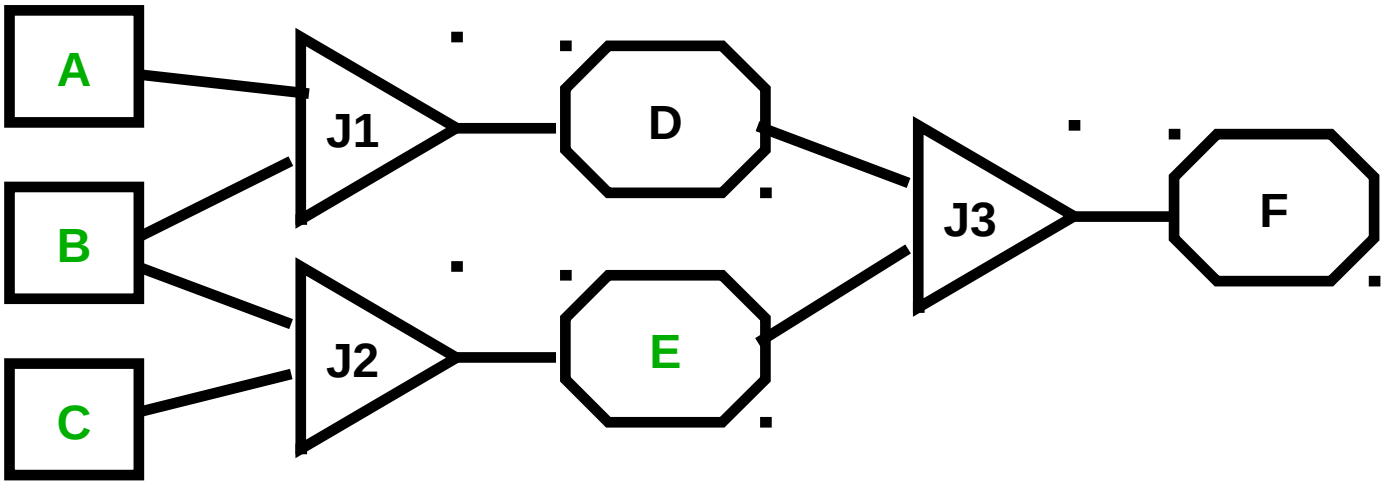
Propagation of Belief

Initial state of dependency network:



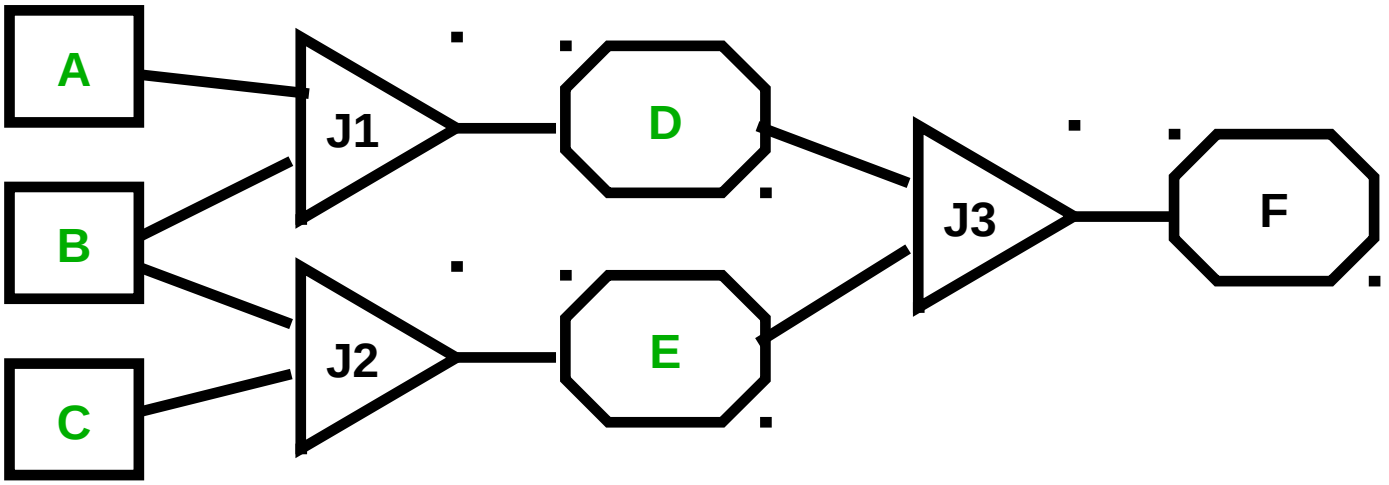
Example

Suppose inference engine enables A:



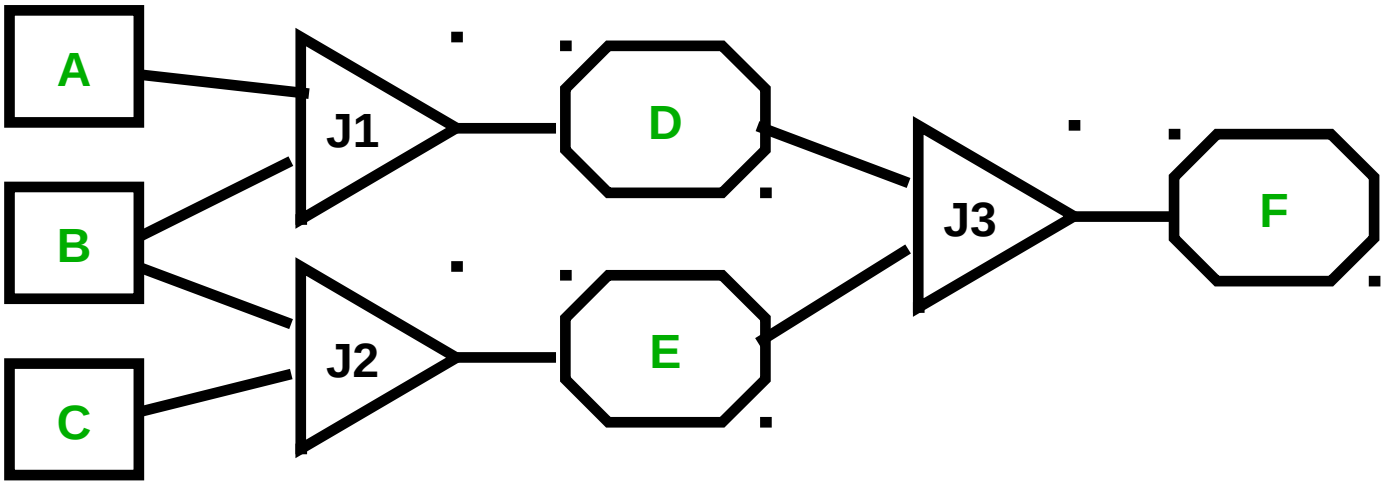
Example

D becomes believed via J1:



Example

F becomes believed via J3:



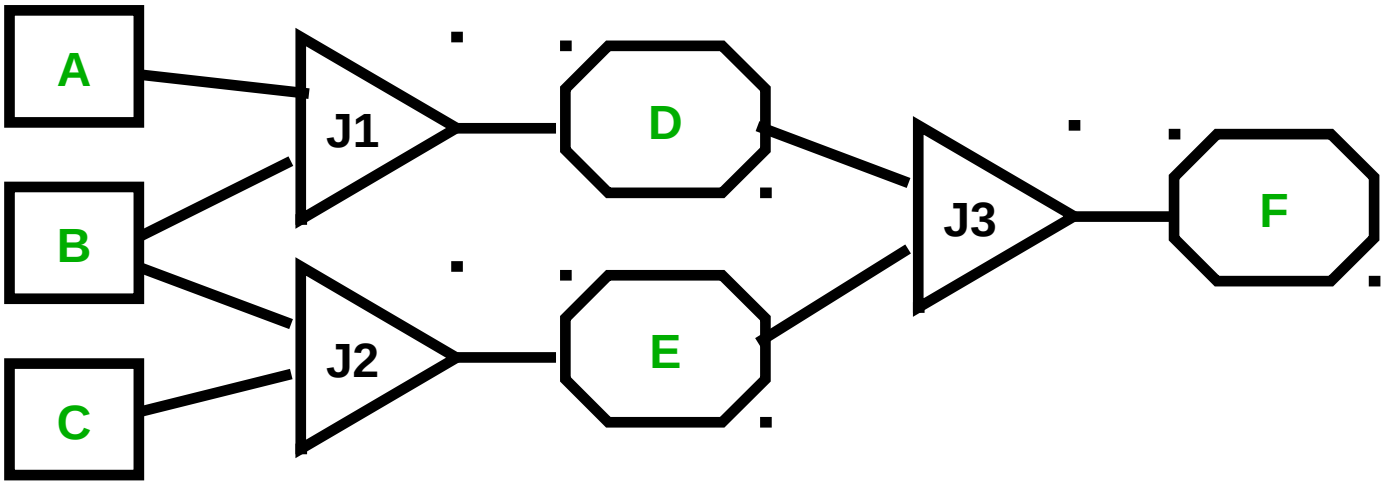
Retracting information

- Premises, contradictions cannot be retracted
- Justifications cannot be retracted
  - They comprise the problem solver's cache
  - Rules need only be run once for each set of matching data
- Assumptions can be retracted
- Algorithm:

1. Make assumption OUT
2. Retract all nodes which rely on it (`Propagate-outness`)
3. Find alternate support for newly OUT nodes.

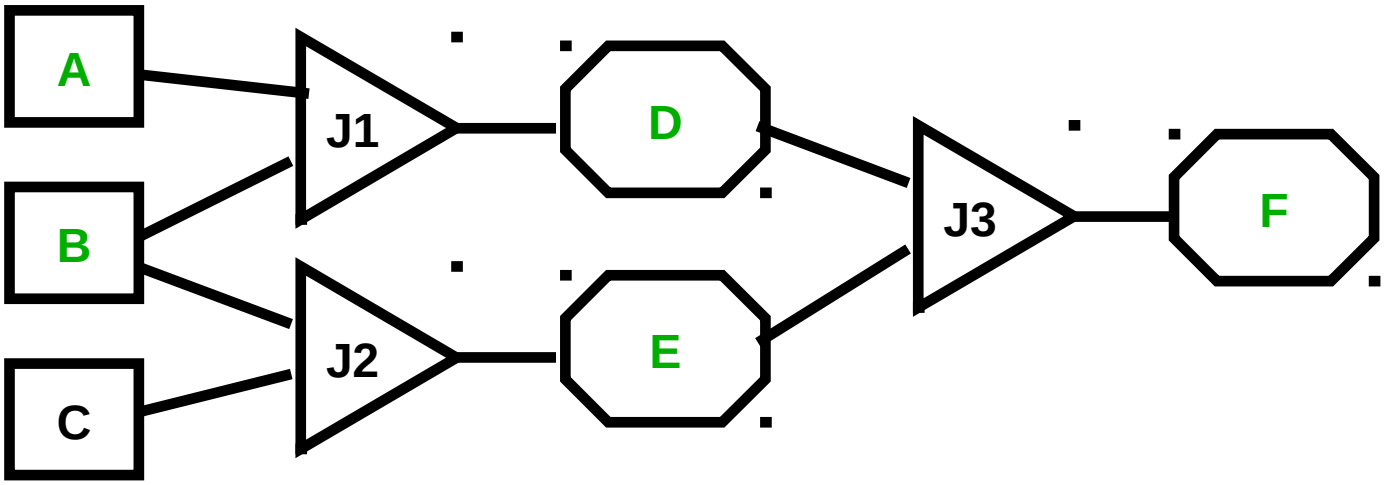
Retraction Example

Initial state:



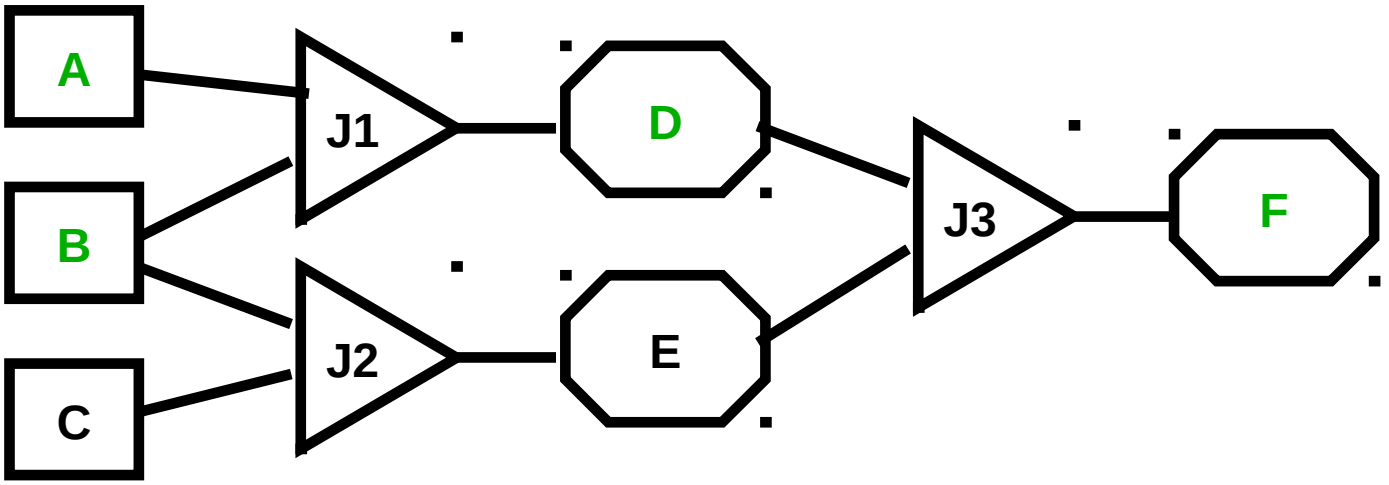
Retraction Example

Retract C:



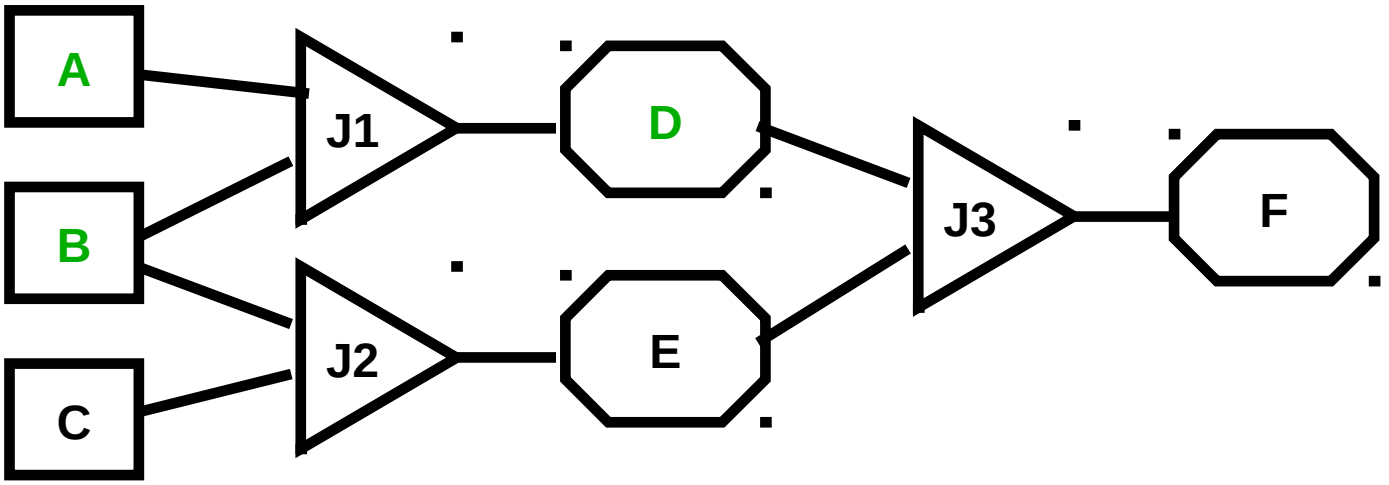
Retraction Example

E becomes out:



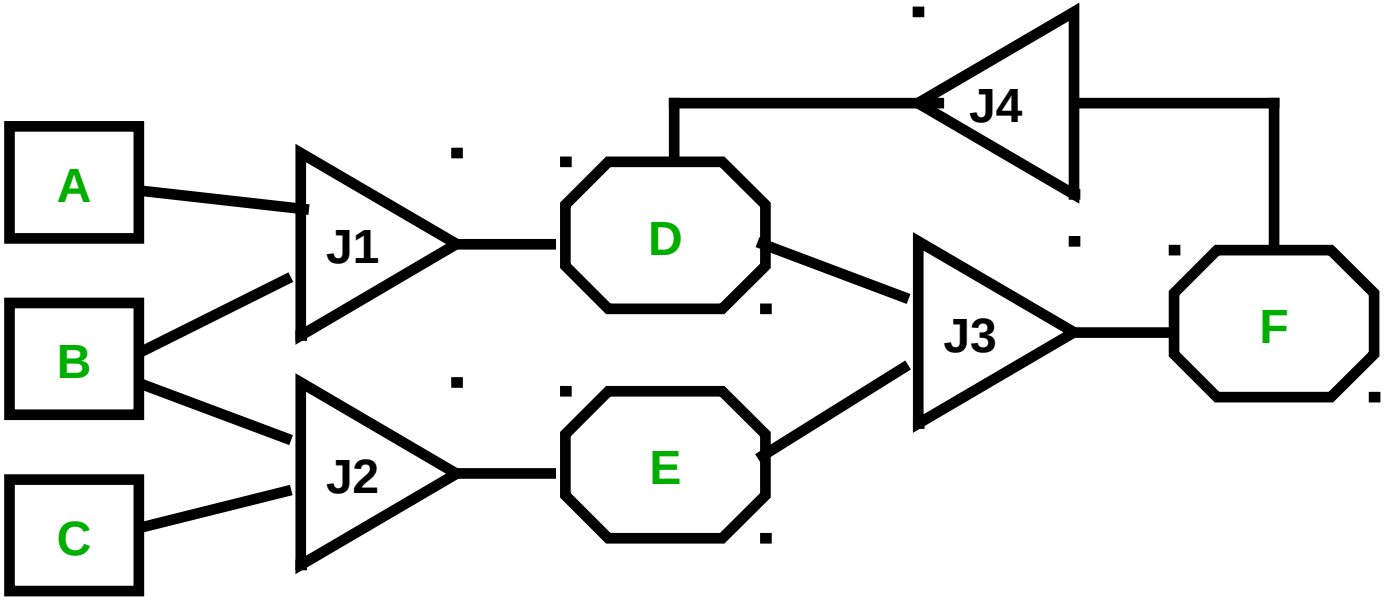
Retraction Example

F becomes out:



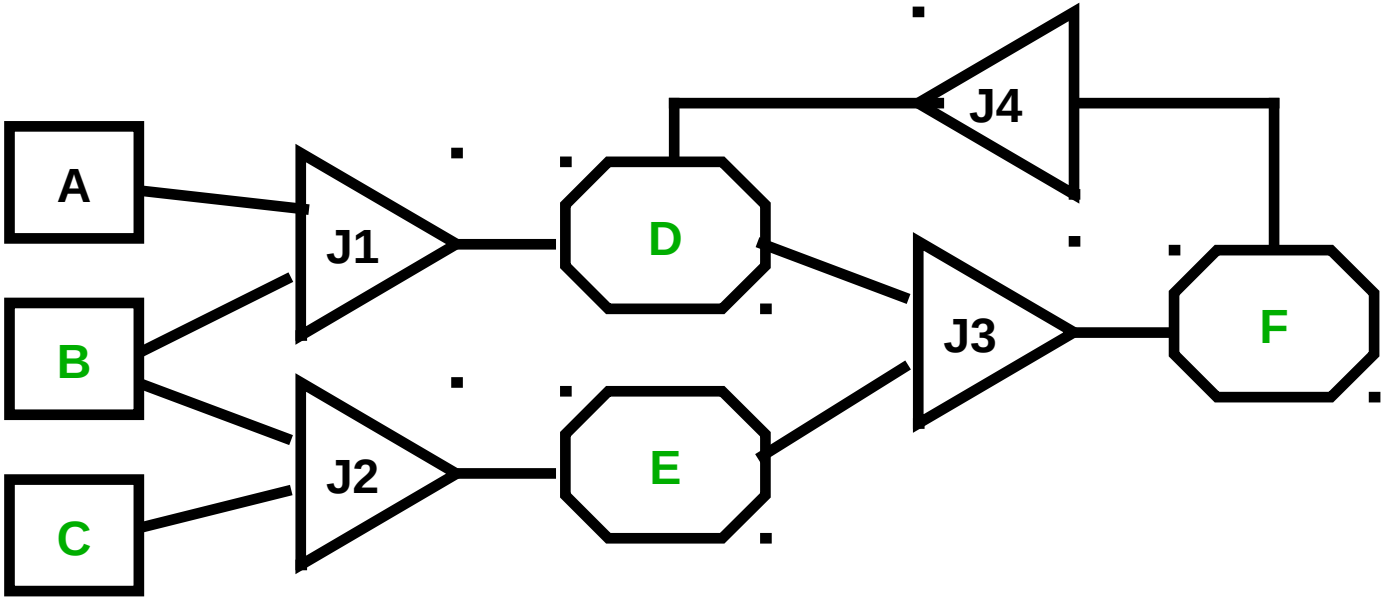
Retract then Resupport

Initial state:



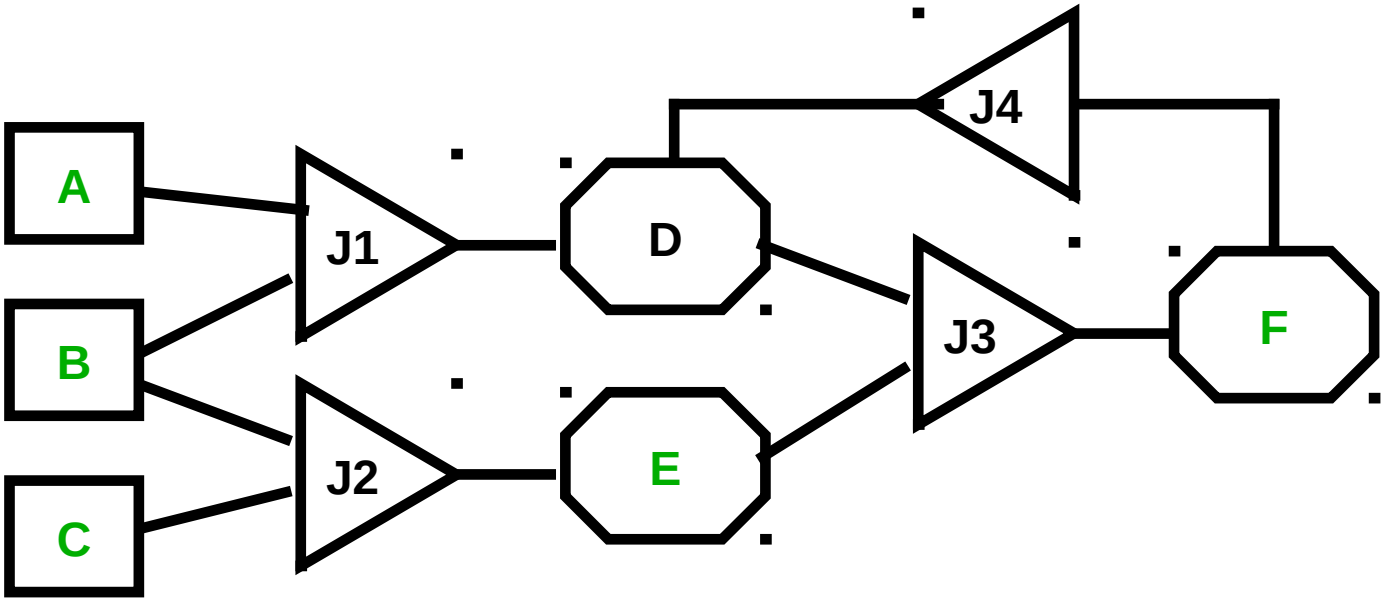
Retract then Resupport

Retract A:



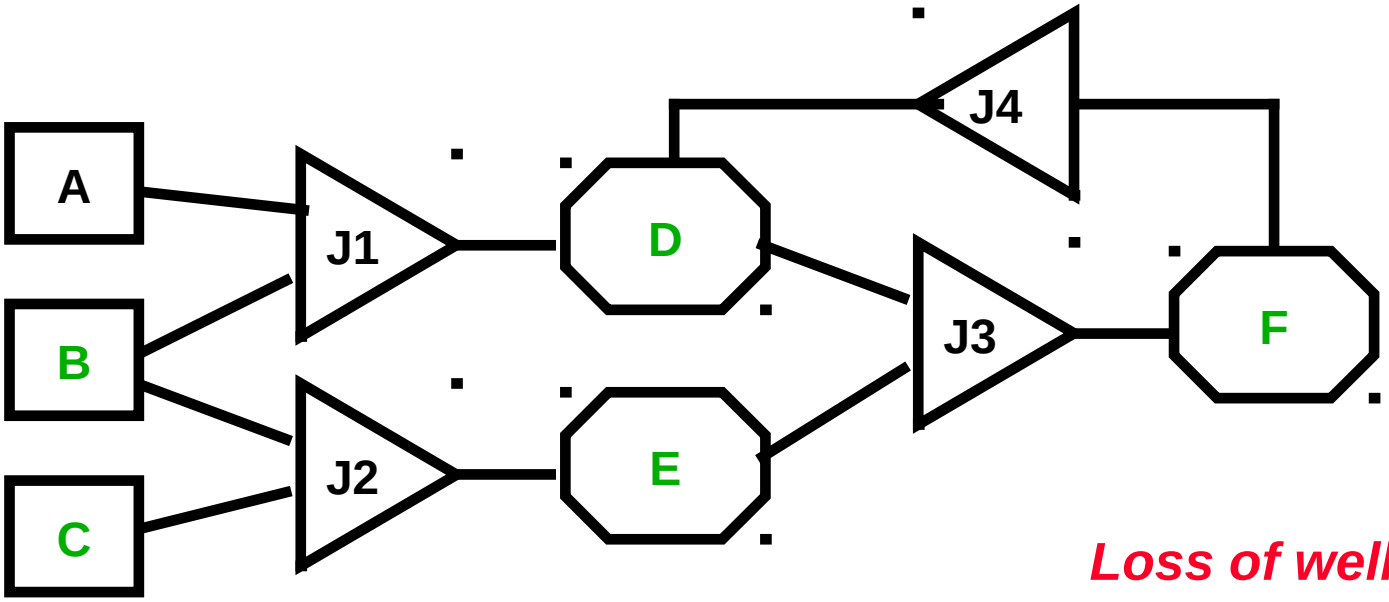
Retract then Resupport

Retract D via J1:



Retract then Resupport

Resupport D via J4:



Loss of well-founded support!

References

- R. Reiter (1980). A logic for default reasoning. Artificial Intelligence, 13:81-132.
- Stuart C.Shapiro, Belief Revision and TMSs: An Overview and a Proposal CSE Technical Report 98-10.
- J.Doyle, A truth maintenance system. Artificial Intelligence, Volume 12, Issue 3, 1979, Pages 231-272.
- Johan de Kleer, An assumption-based truth maintenance system Artificial Intelligence, Volume 28, Issue 2, 1986, Pages 127-162.

TOOLS

- Truth-teller: An assumption based truth maintenance system (ATMS) in C#:

  https://github.com/maate/truth-teller

- JTMS: A simple implementation of Doyle's Justification-based Truth Maintenance System (JTMS):

  https://github.com/hbeck/jtms

- ATMS-in-Python: Assumption-based truth maintenance systems implementation in Python 3 accordingly to the work of Johan de Kleer:

  https://github.com/FellnerDotDev/ATMS-in-Python

The End

Byebye!
See you at the exam!
I will be there!