Logic-based Truth Maintenance System (LTMS)

- Incrementally maintains consequences of a propositional theory Σ
  - incrementally manages addition and deletions from Σ
- Σ is a set of propositional clauses
  - a *clause* is disjunction of propositional literals
  - a *unit clause* is a clause with exactly one disjunct
  - a *literal* is a proposition or the negation of a proposition

$$\neg\, \mathit{rain} \qquad \neg\, \mathit{umbrella} \qquad \mathit{dry}$$

  - a clause can be read as an implication in different ways

$$\mathit{rain} \wedge \mathit{umbrella} \qquad \mathit{dry}$$

$$\mathit{rain} \wedge \neg\, \mathit{dry} \qquad \neg \mathit{umbrella}$$

Generic LTMS interface

- Updating the clauses in Σ
  - *add-clause* (*clause*, Σ)
  - *delete-clause* (*clause*, Σ)
- Propositional inference
  - *consistent?* (Σ)
  - *follows-from?* (*literal*, Σ)
- Justification structure
  - *supporting-clause* (*literal*, Σ)
  - *supporting-literals* (*literal*, Σ)
    - the *supporting-clause* together with the *supporting-literals* entail *literal*
    - each literal in *supporting-literals* follows from Σ
    - is a special literal denoting a contradiction

Using the LTMS in diagnosis

- LTMS database Σ contains clauses describing component behavior in each mode (*SD*)
- Search algorithm adds and deletes clauses corresponding to assumptions that a component is in a particular mode
  - checks that Σ is consistent
  - justification structure is used to generate conflicts from an inconsistent Σ

LTMS labels

- The LTMS *labels* each proposition *true*, *false*, or *unknown*
  - if p is labeled *true* (*false*), then Σ logically entails p ($\neg p$)
  - labeling algorithm is *sound*, but not necessarily *complete*

Σ

$\neg p$ $\neg q$ r $\neg s$

s p q

$\neg u$ v u v

Labels

p: true	*u: unknown*
q: true	*v:unknown*
r: true	
s: false	

Conflicting clauses

- A *conflicting clause* is one in which all literals are labeled *false*
  - $\neg p \quad \neg q \quad r$ is a conflicting clause if the labels are *p: true, q: true, r: false*
- Existence of a conflicting clause means that Σ is *inconsistent*
- If Σ is inconsistent, *supporting-clause*(, Σ) returns a conflicting clause and *supporting-literals*(, Σ) returns the set of literals in that clause

Unit propagation at the *fringe*

- Unit propagation takes place at the *fringe*, which consists of all clauses that have
  - exactly one literal labeled *unknown*
  - all other literals labeled *false*
- Basic unit propagation algorithm
  - select a clause from the *fringe* and propagate until the *fringe* is empty or a conflicting clause is detected

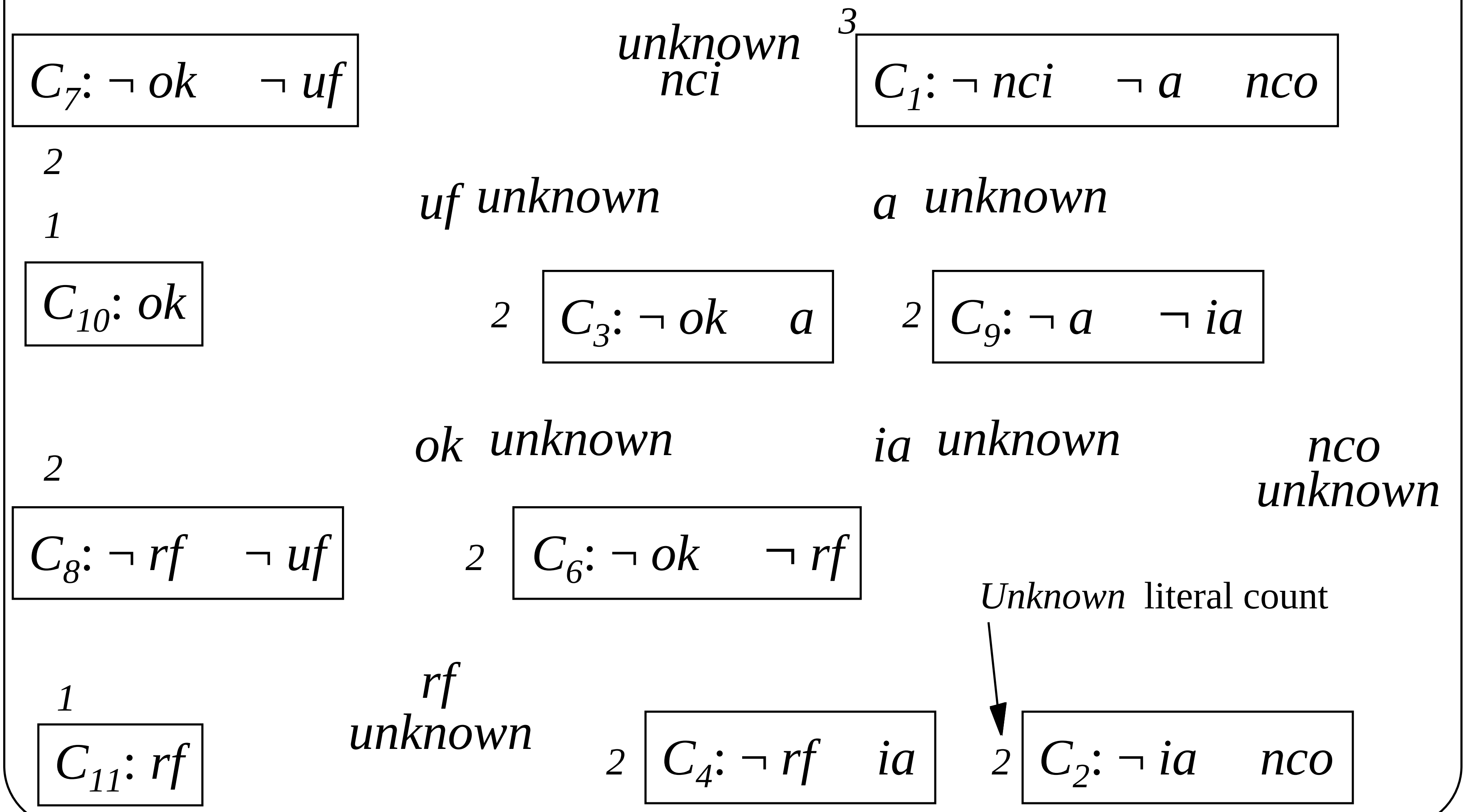
Updating *fringe* and *conflicts*

- *fringe* and *conflicts* updated when a proposition’s label changes
  - only clauses in which the proposition occurs can update *fringe* or *conflicts*
- Membership in *fringe* and *conflicts* determined *incrementally*
  - track the count of literals in the clause labeled *unknown*
    - decrement (increment) the count when an *unknown* (*true* or false) literal becomes *true* or *false* (*unknown*)
  - track whether the clause is satisfied (*i.e.*, contains a literal labeled *true*)

A clause is added to (removed from) the *fringe* if the *unknown* literal count becomes (changes from) 1 and it is not (or it is) satisfied

A clause is added to (removed from) the *conflicts* if the *unknown* literal count becomes (changes from) 0 and it is not (or it is) satisfied

LTMS after initialization



$C_7: \neg\, ok \quad \neg\, uf$

2

1

$C_{10}: ok$

2

$C_8: \neg\, rf \quad \neg\, uf$

1

$C_{11}: rf$

unknown *nci*

3 $C_1: \neg\, nci \quad \neg\, a \quad nco$

uf unknown

a unknown

2 $C_3: \neg\, ok \quad a$

2 $C_9: \neg\, a \quad \neg\, ia$

ok unknown

ia unknown

nco *unknown*

2 $C_6: \neg\, ok \quad \neg\, rf$

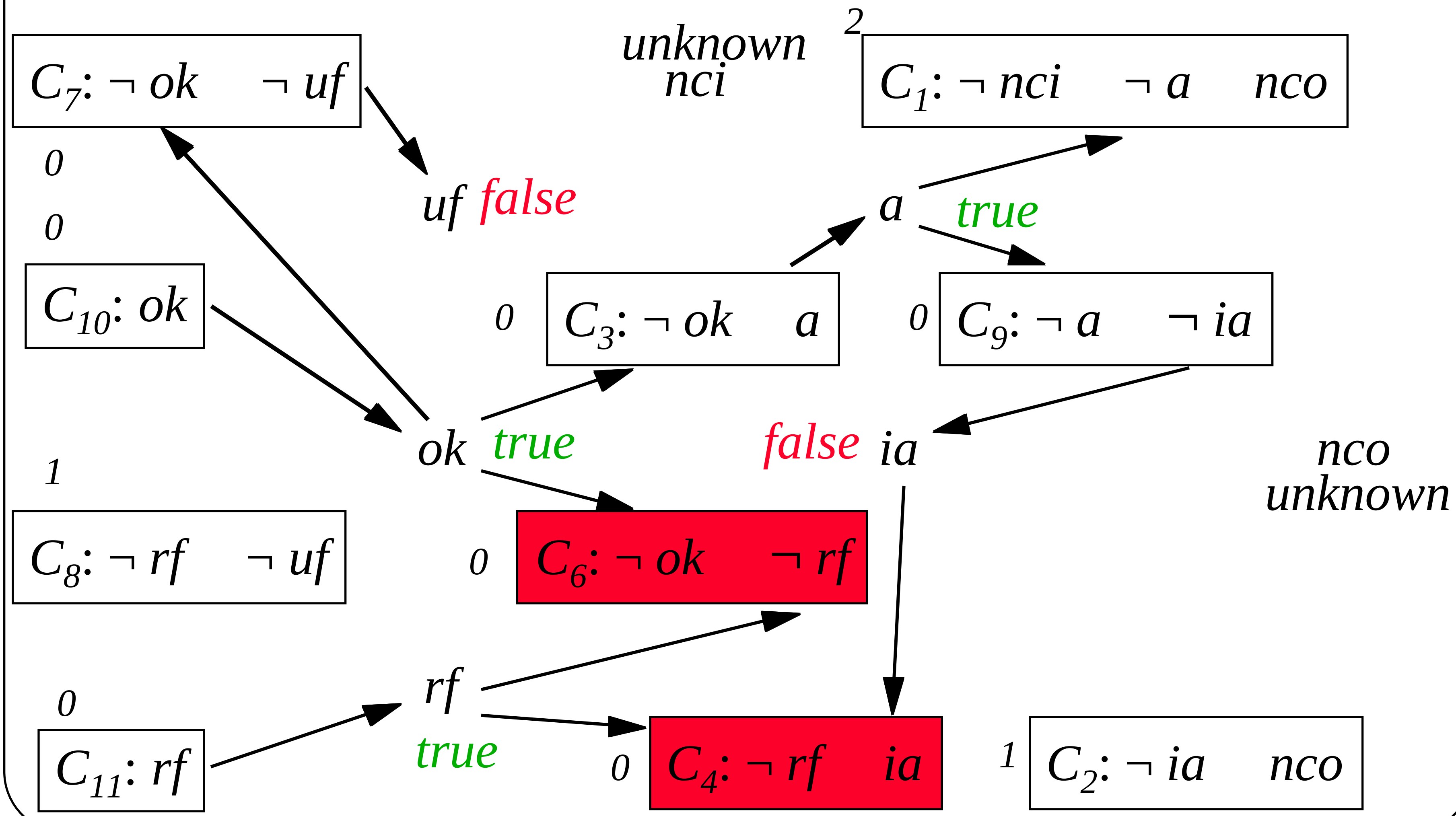
Unknown literal count

rf *unknown*

2 $C_4: \neg\, rf \quad ia$

2 $C_2: \neg\, ia \quad nco$

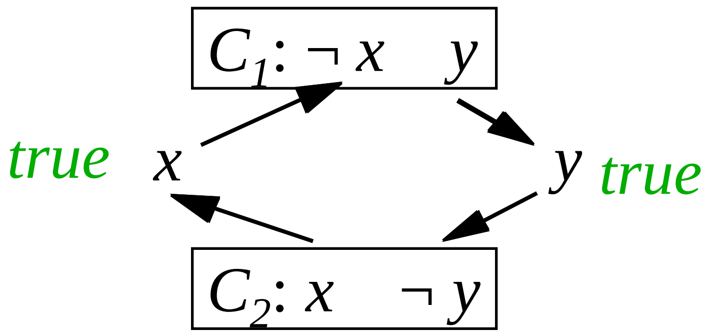
After propagation



Well-founded support

- Proposition supports generated by unit propagation form a *directed acyclic graph*

  Unit propagation produces *well-founded support*
- *Non-well-founded support* contains cycles in the support graph



$C_1: \neg x \quad y$

true x

y *true*

$C_2: x \quad \neg y$

Implementing the generic interface

- *consistent?* (Σ)
  - returns *true* iff Σ has no conflicts after unit propagation terminates
- *follows-from?* (*literal*, Σ)
  - returns *literal*'s label after unit propagation terminates
- *supporting-clause* (*literal*, Σ)

  supporting-literals (*literal*, Σ)
  - returns the clause and literals, respectively, that support *literal* after unit propagation terminates

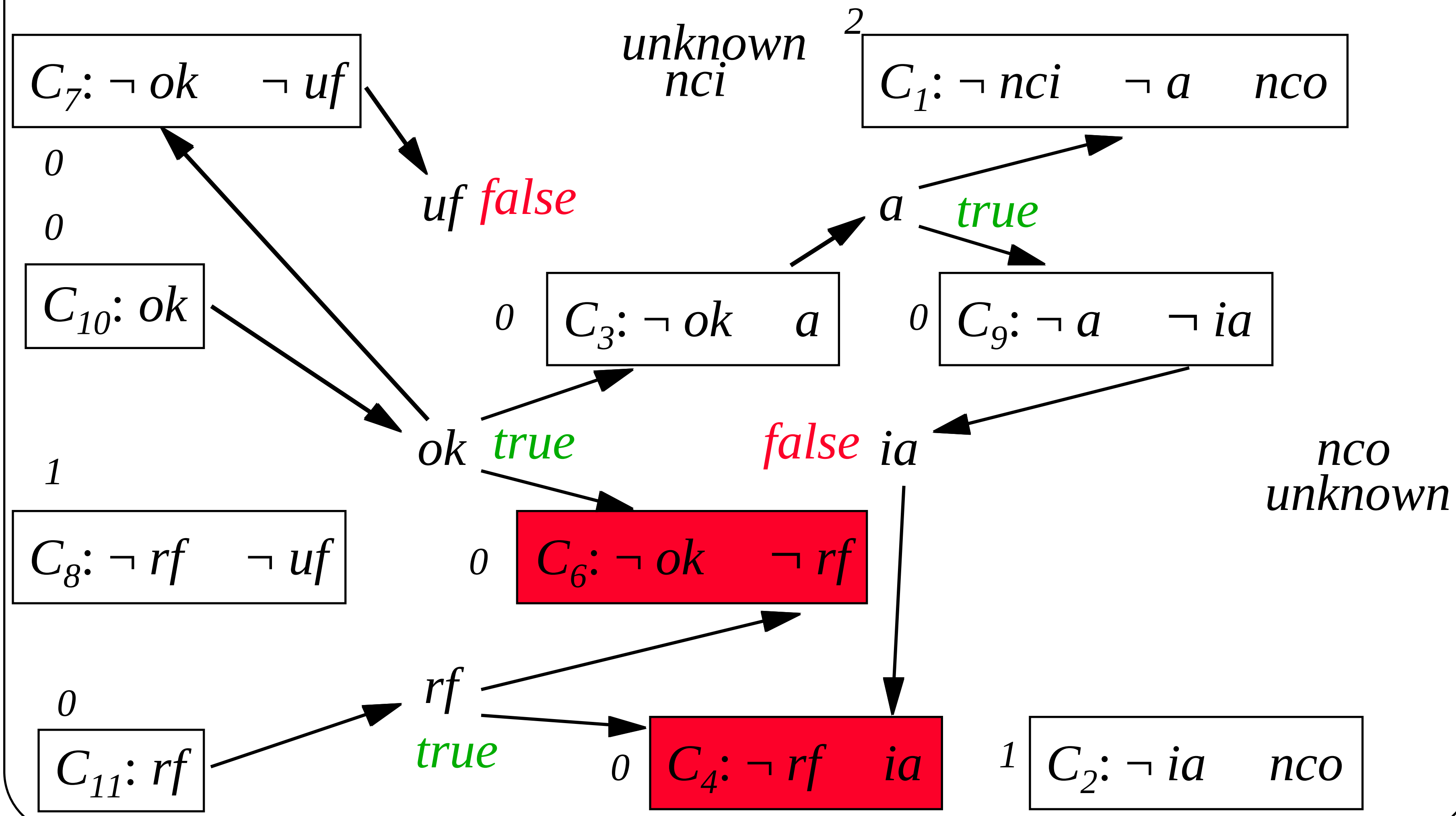
Incrementally modifying Σ

- *add-clause* (*clause*, Σ)
  - update *clause*'s *unknown* literal count and whether it is satisfied
  - update Σ's *fringe* and *conflicts* appropriately
  - call *propagate* (Σ)

    need only do propagations (directly or indirectly) dependent on *clause*
- *delete-clause* (*clause*, Σ)
  - follow the support structure to set the label of all propositions (directly or indirectly) supported by *clause* to *unknown*
    - update Σ's *fringe* and *conflicts* as labels are changed

    only propagations (directly or indirectly) dependent on *clause* are undone
  - call *propagate* (Σ)

Before deleting C_{11}



C_7: ¬ ok ¬ uf

0

0

C_{10}: ok

unknown
nci

2

C_1: ¬ nci ¬ a nco

uf false

a true

0 C_3: ¬ ok a

0 C_9: ¬ a ¬ ia

ok true

false ia

nco
unknown

1

C_8: ¬ rf ¬ uf

0 C_6: ¬ ok ¬ rf

0

C_{11}: rf

rf
true

0 C_4: ¬ rf ia

1 C_2: ¬ ia nco