Theorem Proving - Resolution

Boolean CSP in CNF: Conjunction of clauses
Clause: Disjunction of Literals
Literal: Positive or negative atomic proposition

High-Level Implementation in CHR

Syntax: Clause as *ordered* list of signed variables
For example, $\neg x \vee y \vee z$ as `cl([-x,+y,+z])`.

Resolution with Factoring

```
empty_clause @ cl([]) <=> false.
tautology @ cl(L) <=> in(+X,L),in(-X,L)| true.

cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

Auxiliary predicates:
`in(A,L)`: Element `A` occurs in list `L`.
`del(A,L1,L2)`: List `L1` without element `A` is `L2`.
`merge(L1,L2,L)`: Ordered merge of `L1` and `L2` is `L3`.

Sorry, examples are missing. Try it online with webchr.

Subsumption

Subsumption

```
cl(L1),cl(L2) <=> sublist(L1,L2) | cl(L1).
```

From (Subsumption) and (Resolution)

```
cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

derive Parental Subsumption

```
cl(L1),cl(L2) <=> del(A,L1,L3),del(B,L2,L4),
                  compl(A,B), sublist(L3,L4) |
                  cl(L1),cl(L4).
```

Auxiliary predicates:
`sublist(L1,L2)`: All elements of `L1` occur in `L2`.
`compl(A,B)`: Literal `A` is the logical complement of `B`.

Unit Resolution Davis-Putnam Procedure

From (Subsumption)

```
cl(L1),cl(L2) <=> sublist(L1,L2) | cl(L1).
```

Unit Subsumption

```
cl([A]),cl(L) <=> in(A,L) | cl([A]).
```

From (Parental Subsumption)

```
cl(L1),cl(L2) <=> del(A,L1,L3),del(B,L2,L4),
                  compl(A,B), sublist(L3,L4) |
                  cl(L1),cl(L4).
```

Unit Propagation

```
cl([A]),cl(L) <=> compl(A,B),del(B,L,L1)|
                  cl([A]),cl(L1).
```

Incomplete without search

Ordered Resolution

(Resolution)

```
cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

Ordered Resolution I

```
cl([A|L1]),cl(L2) ==> compl(A,B),del(B,L2,L3)|
                  merge(L1,L3,L),
                  cl(L).
```

Ordered Resolution II

```
cl([A|L1]),cl([B|L2]) ==> compl(A,B) |
                  merge(L1,L2,L),
                  cl(L).
```

Still satisfaction complete

Clause Length Filtering

(Resolution)

```
cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

Filtered Resolution

```
cl(L1),cl(L2) ==> maxlen(L1,n1),maxlen(L2,n2),
                  compl(A,B),
                  del(A,L1,L3),del(B,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

$\texttt{n1} = 1,\ \texttt{n2} = \infty$	Unit propagation
$\texttt{n1} = 1,\ \texttt{n2} = 1$	Contradiction
$\texttt{n1} = 2,\ \texttt{n2} = 2$	Binary Resolution
$\texttt{n1} = 2,\ \texttt{n2} = 3$	Ternary Resolution

Incomplete without search

Search in CHR$^\vee$

Retain completeness for restricted resolution by
Search = Enumeration = Labeling =
Branching = Case Analysis = Choices

Variables

```
enum([X|L]) <=> (cl([+X]) ; cl([-X])),
                 enum(L).
```

Clauses I

```
label, cl(L) <=> in(A,L), cl([A]), label.
```

Clauses II

```
split, cl(L) <=> split(L,L1,L2),
                 (cl(L1) ; cl(L2)),
                 split.
```

The semicolon (;) denotes disjunction implemented by search.

Constraint Inference

Derive equality and partial order constraints.

Equivalence

```
cl([+X,-Y]),cl([-X,+Y]) <=> X = Y.
```

Replace all `X` by `Y`.

Implication

```
cl([+X,-Y]) ==> X >= Y.
```

Helps finding equivalences.

Exclusive Or

```
cl([+X,+Y]),cl([-X,-Y]) <=> X \= Y.
```

Replace all `X` by `-Y`.

Projection by Variable Elimination

(Resolution)

```
cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

Variable Elimination

```
eliminate(X),
cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
eliminate(X),
cl(L) <=> (in(+X,L);in(-X,L)) | true.

eliminate(X) <=> true.
```

Projection by Variable Elimination

Example

```
?- cl([-x,+y]), cl([-x,+z]), cl([+x,-y,-z]),
eliminate(y).

cl([-(x),+(z)])


?- cl([-x,+y]), cl([-x,+z]), cl([+x,-y,-z]),
eliminate(z).

cl([-(x),+(y)])


?- cl([-x,+y]), cl([-x,+z]), cl([+x,-y,-z]),
eliminate(x), eliminate(y).

true

?- cl([-x,+y]), cl([+x]), cl([-y]),
eliminate(x), eliminate(y).

false
```

Full First Order Resolution

(Resolution)

```
cl(L1),cl(L2) ==> del(+X,L1,L3),del(-X,L2,L4)|
                  merge(L3,L4,L),
                  cl(L).
```

First Order Resolution

```
cl(C1),cl(C2) ==> copy(C1,L1),copy(C2,L2),
                  del(A,L1,L3),del(B,L2,L4),
                  compl(A,B),unify(A,B) |
                  merge(L3,L4,L),
                  cl(L).
```