



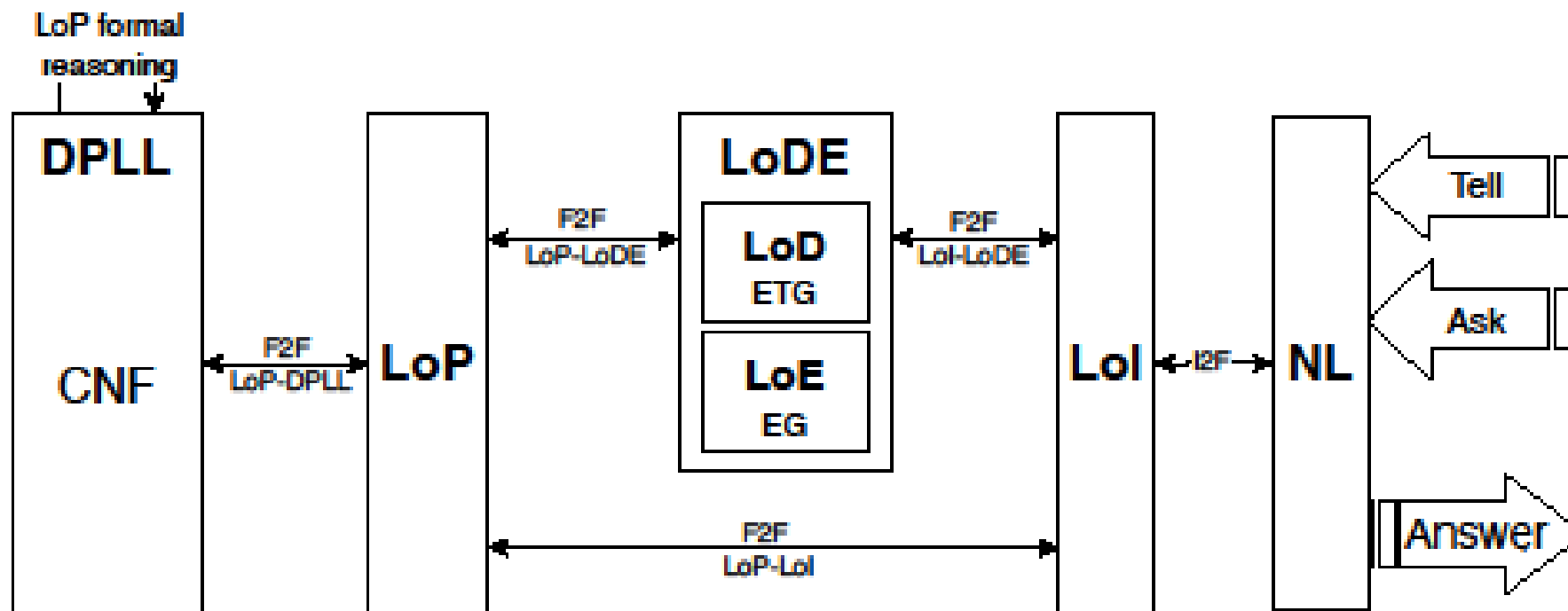
# Reasoning about Propositions

## The big picture (HP2T)

# Reasoning about propositions

- **Intuition**
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - Quantifier elimination
- Lol to ALC / LoDE
- Key notions

# Reasoning about propositions – the big picture



**Notation.** NL: Natural (informal) Language. I2F: Informal to Formal. F2F: Formal to Formal

# Reasoning about propositions

- Intuition
- **Normal Forms**
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - Quantifier elimination
- Lol to ALC / LoDE
- Key notions

# Normal Forms

**Definition (Literal)** A literal is either a proposition or the negation of a proposition, two examples being the formulas  $p$ ,  $\neg q$

**Definition (Negative Normal Form (NNF)).** Any formula involving only disjunctions and conjunctions of literals. A CNF formula has the following generic shape:

$$(L_{\langle 1,1 \rangle} \wedge \dots \vee L_{\langle 1,n_1 \rangle}) \wedge \dots \vee (L_{\langle m,1 \rangle} \vee \dots \vee L_{\langle m,n_m \rangle})$$

**Definition (Conjunctive Normal Form (CNF)).** A conjunction of disjunctions of literals. A CNF formula has the following generic shape:

$$(L_{\langle 1,1 \rangle} \vee \dots \vee L_{\langle 1,n_1 \rangle}) \wedge \dots \wedge (L_{\langle m,1 \rangle} \vee \dots \vee L_{\langle m,n_m \rangle})$$

**Definition (Disjunctive Normal Form (DNF)).** A disjunction of conjunctions of literals. A DNF formula has the following generic shape:

$$(L_{\langle 1,1 \rangle} \wedge \dots \wedge L_{\langle 1,n_1 \rangle}) \vee \dots \vee (L_{\langle m,1 \rangle} \wedge \dots \wedge L_{\langle m,n_m \rangle})$$

# Reasoning about propositions

- Intuition
- Normal Forms
- **Conjunctive Normal Form (CNF)**
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - Quantifier elimination
- Lol to ALC / LoDE
- Key notions

# Conjunctive normal form (CNF)

**Definition (Literal)** A literal is either a propositional variable or the negation of a propositional variable, two examples being the formulas  $p$ ,  $\neg q$

**Definition (Clause).** A clause is a disjunction of literals, an example being the formula  $(p \vee \neg q \vee r)$

**Definition 3 (Conjunctive normal form (CNF)).** A formula is in Conjunctive Normal Form (CNF), if it is a conjunction of clauses, an example being the formula

$$(p \vee \neg q \vee r) \wedge (q \vee r) \wedge (\neg p \vee \neg q) \wedge r$$

# Conjunctive Normal Form (CNF) (Observations)

**Observation (CNF).** A CNF formula has the following shape:

$$(L_{\langle 1,1 \rangle} \vee \dots \vee L_{\langle 1,n_1 \rangle}) \wedge \dots \wedge (L_{\langle m,1 \rangle} \vee \dots \vee L_{\langle m,n_m \rangle})$$

**Example (CNF, special cases).** We have the following limit cases:

- $\{\}$
- $p$
- $\neg p$
- $p \wedge q \wedge r$
- $p \vee q \vee r$





# Properties of clauses 1

**Observation (Order of literals does not matter).** If a clause is obtained by reordering the literals of a another clause, then the two clauses are equivalent.

**Example (Order of literals does not matter).**

$$(p \vee q \vee r \vee \neg r) \equiv (\neg r \vee q \vee p \vee r)$$

**Observation (Order of literals does not matter).** The order does not matter because of the commutativity of disjunction, that is:

$$\phi \vee \psi \equiv \psi \vee \phi$$

# Properties of clauses 2

**Observation (Multiple literals can be merged).** If a clause contains more than one occurrence of the same literal then it is equivalent to the clause obtained by deleting all but one of these occurrences

**Example (Multiple literals can be merged).**

$$(p \vee q \vee r \vee q \vee \neg r) \equiv (p \vee q \vee r \vee \neg r)$$

**Observation (Multiple literals can be merged).** Multiple literals can be merged because of the absorption of disjunction, that is:

$$\phi \vee \phi \equiv \phi$$



# Properties of clauses 3

**Observation (Clauses as sets of literals).** From these properties we can represent a clause as a set of literals, by leaving disjunction implicit and by ignoring replication and order of literals

**Example (Clauses as sets of literals).** The clause

$$(p \vee q \vee r \vee \neg r)$$

can be represented by the set

$$\{p, q, r, \neg r\}$$

# Properties of CNF formulas 1

**Observation (Order of clauses does not matter).** If a CNF formula  $\phi$  is obtained by reordering the clauses of a CNF formula  $\phi'$  then  $\phi$  and  $\phi'$  are equivalent

**Example (Order of clauses does not matter).**

$$(p \vee q) \wedge (r \vee \neg q) \wedge (\neg q) \equiv (r \vee \neg q) \wedge (\neg q) \wedge (p \vee q)$$

**Observation (Order of clauses does not matter).** The order does not matter because of the commutativity of conjunction, that is:

$$\phi \wedge \psi \equiv \psi \wedge \phi$$

# Properties of CNF formulas 2

**Observation (Multiple clauses can be merged).** If a CNF formula contains more than one occurrence of the same clause then it is equivalent to the formula obtained by deleting all but one of the duplicated occurrences

**Observation (Multiple clauses can be merged).**

$$(p \vee q) \wedge (r \vee \neg q) \wedge (p \vee q) \equiv (p \vee q) \wedge (r \vee \neg q)$$

**Observation (Multiple clauses can be merged).** Multiple clauses can be merged because of the absorption of conjunction, that is:

$$\phi \wedge \phi \equiv \phi$$

# Properties of CNF formulas 3

**Observation (A CNF formula can be seen as a set of clauses).** A CNF formula can be represented as a set of sets of literals.

**Observation (A CNF formula can be seen as a set of clauses).** The following CNF formula

$$(p \vee q) \wedge (r \vee \neg q) \wedge (\neg q)$$

can be represented as

$$\{\{p, q\}, \{r, \neg q\}, \{\neg q\}\}$$



# CNF formulas (main properties)

**Proposition (Existence).** Every LoP formula  $\phi$  can be rewritten into Conjunctive Normal Form, written  $CNF(\phi)$ .

**Proposition (Equivalence)**  $\models CNF(\phi) \equiv \phi$

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- **CNF reasoning**
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - Quantifier elimination
- Lol to ALC / LoDE
- Key notions



# Why DPLL SAT

- **LoP SAT:** SAT is a key property, as it amounts to checking whether a certain theory can be instantiated in practice (think, e.g., of scheduling);
- **NP-complete of LoP SAT:** all the NP-complete problems can be encoded in PL SAT;
- **Deduction theorem:** If  $\Gamma, \phi \models \psi$  then  $\Gamma \models \phi \supset \psi$ , (with  $\Gamma$  possibly empty). This allows to reduce checking logical consequence to a PL SAT problem;
- **CNN SAT:** SAT can be reduced to CNF SAT (for Conjunctive Normal form, see later, i.e., conjunctions of disjunctions of possibly negated atomic propositions);
- **Efficiency:** CNF SAT can be implemented very efficiently by exploiting smart **heuristics** (e.g. strategies for selecting the "best" truth assignment);
- **Implementation:** State of the art SAT solvers, called CDCL (for Conflict-Driven Clause-Learning) solve industrial problem up to a few million atomic propositions;
- **Innovation:** Last year, IBM and Google announced a quantum computer solving SAT problems.

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - **CNF Satisfiability**
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - Quantifier elimination
- Lol to ALC / LoDE
- Key notions

# CNF satisfiability – Base

**Proposition (CNF Satisfiability).** Let

$$CNF(\phi) = C_0, \dots, C_n$$

be a formula in CNF, where  $C_0, \dots, C_n$  are the clauses in  $CNF(\phi)$ . Then we have the following:

- $I \models \phi$  if and only if  $I \models C_i$  for all  $i = 0 \dots n$
- $I \models C_i$  if and only if for some literal  $L \in C_i$ ,  $I \models L$

where  $I$  is an interpretation and  $L$  is a literal.

**Observation (CNF Satisfiability).** The above proposition says that a formula  $\phi$  is satisfiable if and only if all the clauses which occur in  $CNF(\phi)$  are satisfiable. In turn a clause is satisfiable if the interpretation function (that is, the model) makes it true.

# Partial evaluation

**Observation (Satisfiability of a formula).** To check if a model  $I$  satisfies a formula  $\phi$  we do not need to know the truth values that  $I$  assigns to all the literals appearing in  $\phi$ .

**Example (Satisfiability of a formula).** For instance, if  $I(p) = T$  and  $I(q) = F$ , we can say that

$$I \models \{\{p, q, \neg r\}, \{\neg q, s\}\}$$

**Definition (Partial evaluation).** A partial evaluation is a partial function that associates to some propositions a truth value, but not to all of them.

**Observation (Partial evaluation).** Let  $I$  be a partial evaluation. Then the literals and clauses occurring in  $I$  can be true, false or undefined. We have the following four cases:

- ✧ A clause is true under  $I$  if at least one of its literals is true;
- ✧ A clause is false (or conflicting) if all literals are false;
- ✧ In all the other cases, a clause  $C$  is undefined (or unresolved).

A clause is left undefined when the truth value of its literals is irrelevant to the formula satisfiability.

# Literal evaluation

**Definition (Formula simplification by positive literal).** For any CNF formula  $\phi$  and proposition  $p$ ,  $\phi|_p$  stands for the formula obtained from  $\phi$  by

- replacing all occurrences of  $p$  by the truth value  $\top$  and
- by simplifying the result by removing:
  - the clauses containing the disjunctive term  $\top$ ;
  - the literals  $\neg\top = \perp$  in all remaining clauses.

**Definition (Formula simplification by negative literal).** For any CNF formula  $\phi$  and proposition  $p$ ,  $\phi|_{\neg p}$  stands for the formula obtained from  $\phi$  by

- replacing all occurrences of  $p$  by the truth value  $\perp$  and
- by simplifying the result by removing:
  - the clauses containing the disjunctive term  $\neg\perp = \top$ ;
  - the literals  $\top$  in all remaining clauses.

# Literal evaluation (example)

**Example (Simplification of a formula by an evaluated literal).** Consider the formula below

$$\{\{p, q, \neg r\}, \{\neg p, \neg r\}\} \mid \neg p = \{\{q, \neg r\}\}$$

The second clause is verified because it contains  $\neg p$  which we assume to be true.

The first clause which contains  $p$  which can be eliminated. So we keep this clause, in a simplified form, and we try to verify it via the evaluation of the remaining literals.

# CNF satisfiability – Final

**Proposition (CNF satisfiability).** Let  $\phi$  be a formula with

$$CNF(\phi) = C_0, \dots, C_n$$

where  $C_0, \dots, C_n$  are the clauses in  $CNF(\phi)$ . Let us assume that we iterate the process of literal evaluation.

Then, the process will terminate with one of two possible situations:

- $\{\}$ , that is, with an **empty set of clauses**, in which case  $\phi$  is satisfiable;
- $\{\dots\{\}\dots\}$ , that is, with a **non empty set of clauses containing one empty clause**, in which case  $\phi$  is unsatisfiable.

# CNF satisfiability (example)

**Example.** Check the satisfiability of the following formula

$$(\neg p \vee q) \wedge (\neg r \vee q)$$

1.  $(\neg p \vee q) \wedge (\neg r \vee q)$
2.  $\{\{\neg p, q\}, \{\neg r, q\}\}$
3.  $\{\{\neg p, \top\}, \{\neg r, \top\}\} \mid q$
4.  $\{\}$



# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - **DPLL**
- LoP to CNF
- LoI to LoP
  - Finite Predicate extension
  - Quantifier elimination
- LoI to ALC / LoDE
- Key notions

# DPLL decision procedure – base

## Algorithm DPLL

Input:  $\varphi = \{c_1, \dots, c_n\}$ .

Output:  $I$ .

Call **DPLL**( $\varphi, \{\}$ )

**DPLL**( $\varphi, I$ )

**if**  $\{\} \in \varphi$

**then exit-return**  $\{\}$  **end**

**if**  $\varphi = \{\}$

**then return**  $I$  **end**

$L \leftarrow$  select-literal( $\varphi$ );

**DPLL**( $\varphi|_L, I \cup \{L\}$ ) **or** **DPLL**( $\varphi|_{\neg L}, I \cup \{\neg L\}$ )

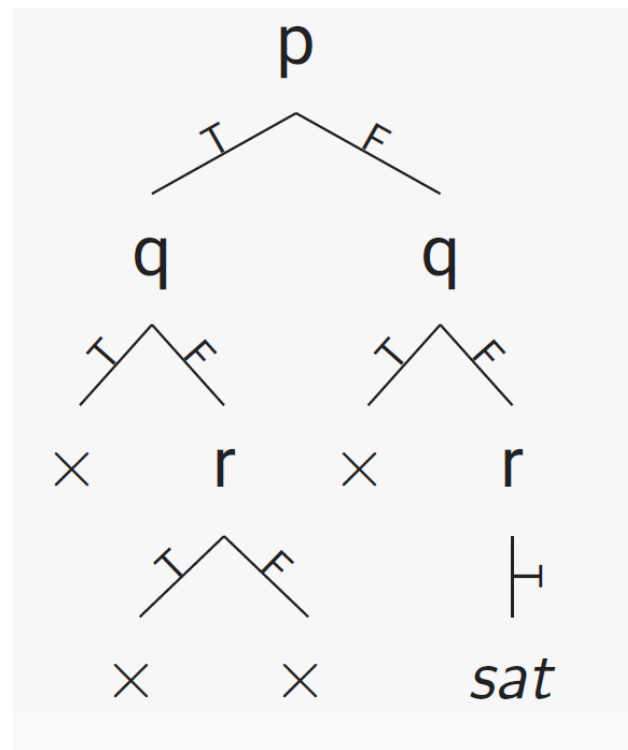
# Backtracking in DPLL (example)

**Observation (backtracking).** The selection of the literal is heuristic. As such, it may lead to a dead-end and generate a need for **backtracking**. Lots of research is focused on advanced heuristics.

**Example (Backtracking).** Consider the following formula:

$$(p \vee \neg q) \wedge (p \vee r) \wedge (\neg p).$$

*The search for an assignment can be represented by the following tree*



# Unit propagation – enhancement 1

**Observation (Unit propagation).** Assume that we have a unit clause in the input formula. How would you modify the algorithm produced in the previous step to take into account this situation. When do you check this information?

**Observation (Unit propagation).** Consider the following examples

1.  $(p \supset q \supset r) \wedge p \wedge \neg q$
2.  $(p \wedge q) \vee \neg p \supset r$
3.  $(p \wedge r) \vee (\neg q \wedge p) \vee (\neg r \wedge \neg p)$

Execute DPLL first without and then with your modification. Then compute how many iterations you saved

# Pure literal – enhancement 2

**Observation (Pure literal).** Assume that a literal occurs only positively or only negatively. How would you modify the algorithm produced in the previous step to take into account this situation. When do you check this information?

**Example (Pure literal).** Consider the following examples

1.  $((p \supset q) \supset r) \wedge p \wedge q$
2.  $((p \wedge q) \supset r) \wedge (p \supset r)$
3.  $(p \wedge \neg r) \vee (q \wedge p) \vee (\neg r \wedge q)$

Execute DPLL first without and then with your modification. Then compute how many iterations you saved

# Literal counting – enhancement 3

**Observation (Literal counting).** Assume that you count the number of time each single literal occurs in a formula. How would you modify the algorithm produced in the previous step to take into account this additional information? When do you compute this information? Is it guaranteed to improve performance?

**Exercise (Literal counting).** Consider the following examples

1.  $(p \supset q \supset r) \wedge p \wedge \neg q$
2.  $(p \wedge q) \vee \neg p \supset r$
3.  $(p \wedge r) \vee (\neg q \wedge p) \vee (\neg r \wedge \neg p)$

Execute DPLL first without and then with your modification. Then compute how much iterations you saved

# DPLL decision procedure – final

**DPLL( $\varphi, I$ )**

**if  $\{\}$   $\in \varphi$  then exit  $\{\}$  end;**

**if  $\varphi = \{\}$  then exit-return  $I$  end;**

**while  $\{L\} \in \varphi$**

**do  $\varphi \leftarrow$  DPLL(unit-propagate ( $\varphi|_L, I \cup \{L\}$ )) end;**

**while pure(L) and  $\{L\} \in \varphi$**

**do  $\varphi \leftarrow$  DPLL(pure-literal-assign ( $\varphi|_L, I \cup \{L\}$ )) end;**

**$L \leftarrow$  select-literal( $\varphi$ );**

**DPLL( $\varphi|_L, I \cup \{L\}$  or DPLL( $\varphi|_{\neg L}, I \cup \{\neg L\}$ ))**

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- **LoP to CNF**
- LoI to LoP
  - Finite Predicate extension
  - Quantifier elimination
- LoI to ALC / LoDE
- Key notions



# Conversion to CNF (1)

**Definition 4 (The CNF function)** Given a PL formula  $\phi$  the function CNF, which transforms  $\phi$  in its CNF form, called  $\text{CNF}(\phi)$  is recursively defined as follows:

$\text{CNF}(p)$	$=$	$p$ if $p \in \mathbf{PROP}$
$\text{CNF}(\neg p)$	$=$	$\neg p$ if $p \in \mathbf{PROP}$
$\text{CNF}(\phi \supset \psi)$	$=$	$\text{CNF}(\neg \phi) \otimes \text{CNF}(\psi)$
$\text{CNF}(\phi \wedge \psi)$	$=$	$\text{CNF}(\phi) \wedge \text{CNF}(\psi)$
$\text{CNF}(\phi \vee \psi)$	$=$	$\text{CNF}(\phi) \otimes \text{CNF}(\psi)$
$\text{CNF}(\phi \equiv \psi)$	$=$	$\text{CNF}(\phi \supset \psi) \wedge \text{CNF}(\psi \supset \phi)$
$\text{CNF}(\neg \neg \phi)$	$=$	$\text{CNF}(\phi)$
$\text{CNF}(\neg(\phi \supset \psi))$	$=$	$\text{CNF}(\phi) \wedge \text{CNF}(\neg \psi)$
$\text{CNF}(\neg(\phi \wedge \psi))$	$=$	$\text{CNF}(\neg \phi) \otimes \text{CNF}(\neg \psi)$
$\text{CNF}(\neg(\phi \vee \psi))$	$=$	$\text{CNF}(\neg \phi) \wedge \text{CNF}(\neg \psi)$
$\text{CNF}(\neg(\phi \equiv \psi))$	$=$	$\text{CNF}(\phi \wedge \neg \psi) \otimes \text{CNF}(\psi \wedge \neg \phi)$

... see next page

# Conversion to CNF (2 – continued)

... where

$$(C_1 \wedge \dots \wedge C_n) \otimes (D_1 \wedge \dots \wedge D_m) \quad (*)$$

is defined as:

$$(C_1 \vee D_1) \wedge \dots \wedge (C_1 \vee D_m) \wedge \dots \wedge (C_n \vee D_1) \wedge \dots \wedge (C_n \vee D_m) \quad (**)$$

with  $C_i$  being a conjunction (possibly a single formula) and  $D_j$  being a disjunction (possibly a single formula).

**Example (special cases).** Rewrite the following special cases of (\*) into their corresponding formulas (\*\*)

- Single formula conjuncts:  $(a \wedge b) \otimes (D_1 \wedge \dots \wedge D_m)$
- Single formula disjuncts:  $(C_1 \wedge \dots \wedge C_n) \otimes (a \wedge b)$
- Single formula conjuncts and disjuncts:  $(a \wedge b) \otimes (c \wedge d)$



# Conversion to CNF (example)

**Example (CNF conversion).** Compute the CNF of  $(q \wedge p) \vee \neg p$

$$\begin{aligned} \text{CNF}((q \wedge p) \vee \neg p) &= \\ \text{CNF}((q \wedge p)) \otimes \text{CNF}(\neg p) &= \\ (\text{CNF}(q) \wedge \text{CNF}(p)) \otimes \neg p &= \\ (q \wedge p) \otimes \neg p &= \\ (q \vee \neg p) \wedge (p \vee \neg p) & \end{aligned}$$

# CNF conversion (example)

**Example (Exponential explosion of a CNF conversion).** Try computing the CNF of the following formula

$$p1 \equiv (p2 \equiv (p3 \equiv (p4 \equiv (p5 \equiv p6))))).$$

The formula resulting from the first conversion step is:

$$CNF(p1 \supset (p2 \equiv (p3 \equiv (p4 \equiv (p5 \equiv p6)))) \wedge CNF((p2 \equiv (p3 \equiv (p4(p5 \equiv p6)))) \supset p1)$$

This formula is double the length of the previous formula. Continuing the expansion, the formula will keep growing exponentially.

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- **LoI to LoP**
  - **Finite Predicate extension**
  - Quantifier elimination
- LoI to ALC / LoDE
- Key notions

# Finite predicate extension

**Definition (Finite predicate extension).** A predicate has a finite extension when it holds is true only for a finite set of constants, that is,

$$\forall x. (P(x) \equiv (x = c_1 \vee \dots \vee x = c_n))$$

**Example (Finite predicate extension).** The days of the week are: Monday, Tuesday, ..., Sunday. This can be formalized as the following formula.

$$\forall x. (WeekDay(x) \equiv x = Mon \vee x = Tue \vee \dots \vee x = Sun))$$

**Observation (Finite predicate extension).** This property allows to eliminate a universal quantifier reducing a quantified formula into a ground formulas which can be then reasoned about in LoP. It applies always, also with infinite domains.

**Observation (Finite predicate extension in AI).** When defining data properties, many predicates have this property, all those which enumerate possible values, for instance: phoneNumber, Age, height, distance. Some approximation may be needed.

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - **Quantifier elimination**
- Lol to ALC / LoDE
- Key notions



# Finite domains

**Definition (Finite domain).** A domain is said to be **finite** if it contains a finite amount of units.

**Observation (Finite domains, in AI).** Most AI applications have finite domains. The only issue is a careful management of data types. The trick is, in the translation from LoDE to LoP, to translate the single values of data properties. As before, some approximation maybe necessary. This applies in particular to the modeling of space and time.

**Observation (Finite domains, in CS).** It is a property which is quite extensively exploited in formal methods, as it allows to validate systems using model checking and satisfiability. It does not generalize to programs with loops and recursion.



# Quantifier expansion

**Proposition (Quantifier expansion).** With finite domains, a quantified LoI formula can be grounded to an **equivalent** LoP formula based on the following equivalence:

$$\phi_D = \{c_1, \dots, c_n\} \models \forall x. \phi(x) \quad \equiv \quad \phi(c_1) \wedge \dots \wedge \phi(c_n)$$

$$\phi_D = \{c_1, \dots, c_n\} \models \exists x. \phi(x) \quad \equiv \quad \phi(c_1) \vee \dots \vee \phi(c_n)$$

**Observation (Quantifier expansion).** A universally (existentially) quantified formula can be substituted with a conjunction (disjunction) with as many conjuncts (disjuncts) as there are constants in the language.

**Observation (Quantifier expansion).** The LoP search space grows exponentially with the size of the domain.

# Multi – quantifier expansion

**Definition (Multi-quantifier expansion).** With finite domains, a Lol formula with nested quantifiers can be grounded to an **equivalent** LoP formula based on the following equivalence:

$$\begin{aligned} \phi_D = \{c_1, \dots, c_n\} \models \forall x_1, \dots, x_k. \phi(x_1, \dots, x_k) &\equiv \bigwedge_{c_{i,1}, \dots, c_{i,k} \in \{c_1, \dots, c_n\}} \phi(c_{i,1}, \dots, c_{i,k}) \\ \phi_D = \{c_1, \dots, c_n\} \models \exists x_1, \dots, x_k. \phi(x_1, \dots, x_k) &\equiv \bigvee_{c_{i,1}, \dots, c_{i,k} \in \{c_1, \dots, c_n\}} \phi(c_{i,1}, \dots, c_{i,k}) \end{aligned}$$

**Observation (Multi-quantifier expansion).** A universally (existentially) quantified formula can be substituted with a conjunction (disjunction) with as many conjuncts (disjuncts) **as there are combinations of constants in the language** (all possible combinations).

**Observation (Explosion of quantifier expansion).** Grounding formulas is very expensive because of the combinatorial explosion of all the possible configurations. There are  $n^m$  combinations with  $n$  the number of variables, and  $m$  the size of the domain.

# Quantifier expansion – example

**Example (Quantifier expansion)** The Natural Language formula

"If someone is noisy, everyone is annoyed"

is translated in Lol as the sentence

$$\exists x.(Noisy(x) \supset \forall y.(Annoyed(y))).$$

Assuming that there are three people, that is

$$D=\{\text{Marco, Francesco, Pierre}\}$$

we can ground the above formula as:

$$\begin{aligned} & (Noisy(\text{Marco}) \supset (Annoyed(\text{Marco}) \wedge Annoyed(\text{Francesco}) \wedge Annoyed(\text{Pierre}))) \vee \\ & (Noisy(\text{Francesco}) \supset (Annoyed(\text{Marco}) \wedge Annoyed(\text{Francesco}) \wedge Annoyed(\text{Pierre}))) \vee \\ & (Noisy(\text{Pierre}) \supset (Annoyed(\text{Marco}) \wedge Annoyed(\text{Francesco}) \wedge Annoyed(\text{Pierre}))) \end{aligned}$$

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- LoI to LoP
  - Finite Predicate extension
  - Quantifier elimination
- **LoI to ALC / LoDE**
- Key notions

# ALC translation to Lol – phase 1

**Definition (Lol translation to LoE/LoD/LoDE).** Consider the following two translation functions.

$$\Pi_x(A) = A(x)$$

$$\Pi_x(C \sqcap D) = \Pi_x(C) \wedge \Pi_x(D)$$

$$\Pi_x(C \sqcup D) = \Pi_x(C) \vee \Pi_x(D)$$

$$\Pi_x(\exists r.C) = \exists y.r(x,y) \wedge \Pi_y(C)$$

$$\Pi_x(\forall r.C) = \forall y.r(x,y) \supset \Pi_y(C)$$

$$\Pi_y(A) = A(y)$$

$$\Pi_y(C \sqcap D) = \Pi_y(C) \wedge \Pi_y(D)$$

$$\Pi_y(C \sqcup D) = \Pi_y(C) \vee \Pi_y(D)$$

$$\Pi_y(\exists r.C) = \exists x.r(y,x) \wedge \Pi_x(C)$$

$$\Pi_y(\forall r.C) = \forall x.r(y,x) \supset \Pi_x(C)$$

**Observation (ALC vs LoE/LoD/LoDE).** ALC is an extension of LoE/LoD/LoD. It is the world logic which is deductiovaly equivalent to LoP.

**Observation (Lol translation to LoE/LoD/LoDE).** The intuition is to add the variable implicit in LoD formulas thus aligning it to the language of LoE/ Lol.

# ALC translation to Lol – phase 2

Let  $T$  and  $A$  be a TBox and an ABox, respectively. Let  $\varphi[x/a]$  the formula obtained from  $\varphi$  by replacing all free occurrences of  $x$  with  $a$ . Then, the translation  $\Pi(T)$  and  $\Pi(A)$  of the TBox and the ABox are, respectively:

$$\Pi(T) = \forall x. \bigwedge_{C \sqsubseteq D \in T} (\Pi_x(C) \supset \Pi_x(D))$$

$$\Pi(A) = \bigwedge_{C(a) \in A} \Pi_x(C) [x/a] \quad \wedge \quad \bigwedge_{r(a,b) \in A} r(a,b)$$

**Observation (mapping).** The mapping preserves the intended semantics.

# ALC vs. LoI – Main result

**Theorem (LoE/LoD/LoDE vs. LoI).** Let  $T$  be a TBox,  $A$  an ABox and  $EB = (T, A)$  be a LoDE theory. Let  $C, D$  possibly composite etypes and  $e$  an entity. Then we have the following:

- $EB = (T, A)$  is satisfiable if and only if

$\Pi(T) \wedge \Pi(A)$  is satisfiable;

- $C \sqsubseteq_T D$  if and only if

$(\Pi(T) \supset \forall x. (\Pi_x(C) \supset \Pi_x(D)))$  is valid;

- $b$  is an instance of  $C$  with respect to  $EB = (T, A)$  if and only if

$(\Pi(T) \wedge \Pi(A)) \supset \Pi_x(C) [x/a]$  is valid;

# ALC vs LoI – Example

**Example (LoDE to LoI).** Consider the following LoDE theory

$$\exists \text{attends}. T \sqsubseteq \text{Person}$$
$$\text{Teacher} \equiv \text{Person} \sqcap \exists \text{teaches}. \text{Course}$$
$$\text{Teacher}(\text{Mary})$$

which makes the following assertions

«Whoever attends classes is a person»

«Teachers are persons who teach course»

«Mary is a teacher»



# ALC vs Lol – Example

**Example (LoDE to Lol).** The following LoDE theory

$$\exists \text{attends}. T \sqsubseteq \text{Person}$$
$$\text{Teacher} \equiv \text{Person} \sqcap \exists \text{teaches}. \text{Course}$$
$$\text{Teacher}(\text{Mary})$$

can be translated into the following Lol theory.

$$\forall x. (\exists y. \text{attends}(y, x) \supset \text{Person}(x))$$
$$\forall x. (\text{Teacher}(x) \equiv \text{Person}(x) \wedge \exists y. (\text{teaches}(x, y) \wedge \text{Course}(y)))$$
$$\text{Teacher}(\text{Mary})$$

# Reasoning about propositions

- Intuition
- Normal Forms
- Conjunctive Normal Form (CNF)
- CNF reasoning
  - CNF Satisfiability
  - DPLL
- LoP to CNF
- Lol to LoP
  - Finite Predicate extension
  - Quantifier elimination
- Lol to ALC / LoDE
- **Key notions**

# DPLL – key notions

- Normal forms: NNF, CNF, DNF
- Literal, clause, CNF
- Conversion to CNF
- CNF satisfiability
- Partial evaluation
- Literal evaluation
- DPLL decision procedure
- Backtracking
- Unit propagation
- Pure literal
- Literal counting
- Finite predicate extension
- Finite domain
- Quantifier expansion
- Nested quantifier expansion
- Search space explosion of (nested) quantifier expansion
- Literal evaluation
- DPLL decision procedure
- Backtracking
- Unit propagation
- Pure literal
- Literal counting



# Reasoning about Propositions

## The big picture (HP2T)

# Mapping Logics

1. Intuition
2. LoD definitions from / to LoP (\* Language level translation \*)
3. LoD defs+descrs from / to LoP (\* Knowledge level translation \*)
4. LoDE from / to LoP (\* Data level translation \*)
5. ALC from / to LoP (\* Data level translation \*)
6. Lol from / to LoDE (\* Data level translation \*)
7. Lol from / to LoD Full (\* Knowledge level + prop. translation \*)
8. Lol from / to ALC (\* Data level + prop translation \*)