Automatisation

- Une procédure automatique de résolution consisterait à grandir constamment la base de connaissance en appliquant la règle de la résolution sur chaque paire de clauses possible
  - aussitôt qu'on génère la clause vide, on a réussit à déduire la clause requête
  - lorsqu'il n'est plus possible d'ajouter une nouvelle clause à l'aide de la règle de résolution, on sait qu'on ne peut déduire la clause requête

- Cette procédure pourrait être très lente
  - voir la section 9.5.6 pour des stratégies pour faire des preuves plus efficacement

Applications

- Vérification de logiciel informatique
  - la base de connaissance contient l'information sur l'effet de chaque instructions et leurs conditions pour être exécutées
- Synthèse de logiciel
  - répondre à la question « existe-t-il un programme *p* satisfaisant une certaine spécification »
- Systèmes experts
  - diagnostic médical, automobile : étant donné des « symptômes », quelle est la « maladie »
  - nécessite qu'un expert mette sous forme logique toutes ses connaissances
- Preuve de théorèmes mathématiques

Traiter l'égalité

- La logique du premier ordre inclue normalement la notion d'égalité (indiquée par le symbole « = ») entre les termes
- Une façon de gérer l'égalité est de la définir avec plusieurs formules qui décrivent le concept d'égalité (symétrie, transitivité, etc.)
  - on peut alors utiliser la preuve par résolution
- Pour d'autres façons de gérer l'égalité, voir la section 9.5.5 du livre

- Certains systèmes logiques utilisent **la supposition des noms uniques** (***unique-names assumption***) :
  - deux constantes ayant un symbole différent sont en fait des entités différentes
  - on ne désignera pas une même personne sous deux noms différents

Domaine ouvert vs. fermé

- La logique du premier ordre suppose aussi **un domaine ouvert**
  - il n'y a pas de limite connue sur l'ensemble des objets
- Dans ce cas, on ne peut pas remplacer la quantification universelle par une conjonction très longue
  - ex. : $\forall x\ \textit{personne}(x) \rightarrow \textit{mortel}(x)$ veut dire que toutes les personnes **qui ont existées ou vont exister** sont mortelles

- Certains logiciels de raisonnement logique vont plutôt supposer un domaine fermé
  - on doit alors définir explicitement l'ensemble des objets de notre « monde »
  - une longue conjonction exhaustive et la quantification universelle sont alors équivalentes
- Un raisonnement similaire s'applique pour la quantification existentielle

Monde ouvert vs. fermé

- La logique de premier ordre suppose aussi **un monde ouvert**
  - si un $p(x)$ n'est pas dans la base de formules, ceci n'implique pas que $\neg p(x)$ est vraie

- Là encore, certains logiciels de raisonnement logique vont plutôt supposer le contraire, c.-à-d. un domaine fermé

- **Prolog** est un exemple de langage logique qui suppose
  - **noms uniques** (***unique-names assumption)***
  - **domaine fermé** (***closed-world assumption***)
  - **monde fermé** (***domain closure assumption***)

- Ces suppositions sont appelées **sémantiques des bases de données**

Satisfiabilité

- On a supposé que la base de connaissance initiale ne contenait pas de contradictions
  - ex. : elle ne contient pas $p(x)$ et $\neg p(x)$
  - si une conjonction de clauses (comme $p(x) \wedge \neg p(x)$) ne peut jamais être vraie, on dit qu'elle n'est pas **satisfaisable**
  - déterminer la satisfiabilité d'une conjonction de clauses est un problème NP-complet en général
    - » premier problème NP-complet ayant été découvert
    - » **3-SAT** : le problème de déterminer si une conjonction de clauses de 3 littéraux chacune est satisfaisable