Revision Notes for CO231 Artificial Intelligence

Spring 2018

1 Search

Requirements

1. **Observable**: current state is known.
2. **Discrete**: finitely many next states from each state.
3. **Deterministic**: each action has exactly one outcome.
4. **Known**: possible actions and next states known.

Formal Definition

1. Initial state (**start**).
2. Transition model between states (**graph**).
3. Goal tests (set of **goal** states).
4. Path cost (sum of positive step costs).

---

1. Solution is **path** from initial state to a goal state.
2. Optimal solution has lowest cost.

Algorithms Generate a search tree:

1. Initialise the tree with initial state as root.
2. Repeat:
   (a) If frontier of tree is empty then fail.
   (b) Choose and remove leaf node L from frontier.
   (c) If L is a goal state, then return solution.
   (d) Add L to seen states.
   (e) Expand L (if not seen), adding resulting nodes to frontier.

To add **loop checking**, each seen L is added to an explored set and only expanded if not already seen.

1.1 Uninformed / Blind Search

Choose next unexpanded node using:

1. **Breadth first search**: choose shallowest node.
2. **Uniform cost search**: choose node with lowest path cost.
3. **Depth first search**: choose deepest node.
4. **Depth limited search**: DFS with depth limit l.
5. **Iterative deepening search**: depth limited search with increasing l.

Variants

1. **Backtracking search**: only generate one node when expanded, remember what to generate next.
2. **Bi-directional search**: Two simultaneous searches from start to goal and goal to start.

Properties

1. **Completeness:** do we always find a solution if one exists?
2. **Time complexity**: number of nodes expanded.
3. **Space complexity**: max number of nodes in memory.
4. **Optimality**: do we always find the least-cost solution?

	Completeness	Time Complexity	Space Complexity	Optimality
BFS	Yes if b is finite	$O\left(b^d\right)$	$O\left(b^d\right)$	Yes if costs are constant
Uniform-cost	Yes if b is finite and all costs are positive	$O\left(b^{d+1}\right)$ if costs are constant	$O\left(b^{d+1}\right)$ if costs are constant	Yes
DFS	Yes if s is finite (No without loop checking)	s ($O(b^m)$ without loop checking)	$O\left(bm\right)$	No
Back-tracking DFS	As for DFS but with $O\left(m\right)$ space complexity.			
Depth-limited DFS	No	$s\left(l\right)$ ($O(b^l)$ without loop checking)	$O\left(bl\right)$	No
Iterative deepening DFS	As for BFS but with $O\left(bd\right)$ space complexity.			
Bidirectional BFS	As for BFS but with $O\left(b^{d/2}\right)$ time and space complexity.			

Where:

- b is the maximum branching factor of the search tree.
- d is the depth of the optimal solution.
- m is the max depth of the state space.
- s is the overall size of the state space.
- $s\left(l\right)$ is the size of the state space up to depth l.

1.2 Informed / Heuristic-Based Search

Use a cost estimate to choose node with least-estimated path cost.

1. **Greedy-best-first-search**: cost estimate = heuristic on this node.
2. **A* search**: cost estimate = actual cost to node + heuristic on this node.
3. **Local search**: e.g. genetic algorithms, simulated annealing.

Heuristic Functions Required properties (where $g\left(n,m\right)$ is the actual cost of reaching m from n and $h\left(n\right)$ is the heuristic function applied to n):

1. **Consistent** / **Monotonic**: $h\left(n\right) \leq g\left(n,n'\right) + h\left(n'\right)$ for any n' successor of n. (Doesn't overestimate actual cost to any successor + heuristic applied to that node).
2. **Admissible** (follows from consistency): $h\left(n\right) \leq g\left(n,m_{goal}\right)$ where m_{goal} is the cheapest reachable goal from n. (Doesn't overestimate actual cost to goal).

- Often admissible / consistent heuristics come from a relaxed version of the search problem (more edges).

	Completeness	Optimality
Greedy	Yes if s is finite	No
Greedy (without loop checking)	No	No
A*	Yes if b is finite and all costs are positive	Yes if heuristic is consistent
A* (without loop checking)	Yes if b is finite and all costs are positive	Yes if heuristic is admissible

Properties A* is optimally efficient but often runs out of space.

1.3 Adversarial Search

Used for:

- Competitive environment (opponents with conflicting goals).
- Unpredictable opponent.
- Hard to solve, time limits.

Definitions for two-player, zero-sum games:

- s_0: initial state.
- `PLAYER`$\left(s\right)$: which player moves in a state.
- `ACTIONS`$\left(s\right)$: set of legal moves in a state.
- `RESULT`$\left(s,a\right)$: state resulting from move in a state.
- `UTILITY`$\left(s,p\right)$: win (1), lose (-1) or draw (0).

Optimal strategies for perfect-information, deterministic games:

1. Minimax.
2. Alpha-beta pruning.

```
if ¬TERMINAL(s):
    if PLAYER(s) = MAX then return max_{a=ACTIONS(s)} MINIMAX(RESULT(s, a))
    if PLAYER(s) = MIN then return min_{a=ACTIONS(s)} MINIMAX(RESULT(s, a))
if TERMINAL(s):
    return UTILITY(s)
```

Minimax

Alpha-beta pruning

- α is the minimum value that MAX is guaranteed to achieve so far (minimiser's best guaranteed score).
- β is the maximum value that MAX is a guaranteed to achieve so far (maximiser's best guaranteed score).

Starting with max-value $(s, -\infty, \infty)$:

```
max-value(s, α, β):
    if ¬TERMINAL(s):
        let v = −∞
        for each a in ACTIONS(s):
            let v = max(v, min-value(RESULT(s, a), α, β))
            if v ≥ β then return v
            else let α = max(α, v)
        return v
    if TERMINAL(s):
        return UTILITY(s)
```

```
min-value(s, α, β):
    if ¬TERMINAL(s):
        let v = ∞
        for each a in ACTIONS(s):
            let v = min(v, max-value(RESULT(s, a), α, β))
            if v ≤ α then return v
            else let β = min(β, v)
        return v
    if TERMINAL(s):
        return UTILITY(s)
```

- **Key idea**: If this move ever gives the opponent a choice to do better than the best they could do in a different move, then we shouldn't consider it.

	Completeness	Time Complexity	Space Complexity	Optimality
Minimax	Yes if search tree is finite	$O(b^m)$	$O(bm)$	Yes
a-β **pruning**	As for minimax but with $O(b^{m/2})$ time complexity.			

Properties

Limited Resources

- Cutoff-test instead of TERMINAL.
- Evaluation instead of UTILITY.

2 Planning

Requirements Observable, discrete, deterministic, known.

2.1 Representation

Feature-centric For each feature define:

- **Causal rules**: specify when a feature gets a new value.
- **Frame rules**: specify when a feature keeps its value.

We still need preconditions for actions.

Action-centric For each action define:

- **Preconditions**: features that must be true for an action to occur.
- **Effects**: features that change as a result of the actions.

Assume:

- Unmentioned **primitive** features aren't changed.
- **Derived** features are expressed in terms of primitive features.

STRIPS Language for Planning

1. **States** - conjunctions of ground atoms.
2. **Goals** - conjunctions of literals - possibly containing variables - implicitly existentially quantified.
3. **Operators**:
   (a) Action description.
   (b) **Precondition** - conjunction of literals.
   (c) **Postcondition** (**Effect**) - conjunction of literals. Every variable in effect must appear in pre-condition.
4. **Actions** - fully instantiated operators.

$$\text{Precondition} \xrightarrow{\text{Action}} \text{Postcondition}$$

Execution of an Action New State = Current state - negation of negative fluents in effects + positive fluents in effects

Assumptions

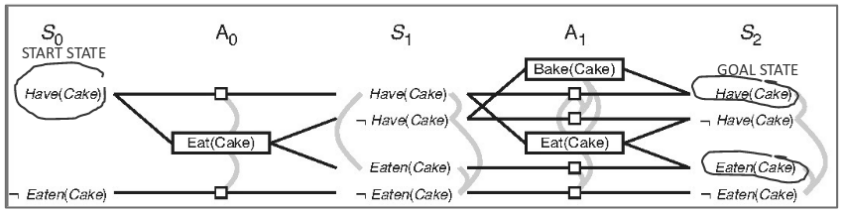
- Fluents not mentioned in a state are false.
- Fluents do not change unless they are explicitly changed by an action.
- Time is implicit.

2.2 Planning as Search

1. **Progression planning** (forward) - blind search with the graph obtained from the specification of the planning problem.
   (a) **Problem**: Large search spaces with many irrelevant actions.
   (b) **Solution**: Good heuristics; relax the problem, add extra edges e.g. by ignoring preconditions and effects not in goal.
2. **Regression planning** (backward) - blind search backwards from the goal (try to achieve sub-goals).
   (a) **Problem**: We may have to search back from many goal states.
   (b) **Problem**: Non-interleaved regression planning is incomplete.

2.3 Graph-Plan Algorithm

Important: don't forget that actions can be done in parallel!



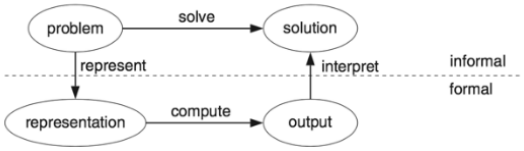
- **Level** S_i:
  - $i = 0$: Start state + literals that could hold at the start.
  - $i > 0$: all literals that could hold at S_i, depending on actions at A_{i-1}.
- **Level** A_i:
  - Each action in A_i connected to its precondition in S_i and effect at S_{i+1}.
  - No-op actions: no change.
- Mutual exclusions between actions when:
  1. Actions have **inconsistent effects**.
  2. **Interference**: effect of one action = ¬ precondition of the other.
  3. **Competing needs**: mutually exclusive pre-conditions.
- Mutual exclusions between literals when:
  1. They are **complementary**.
  2. **Inconsistent support**: Each possible pair of actions achieving them are mutex.
- Levels off: when two consecutive levels are identical.

Extracting the Plan Once goal is non-mutex in S_i, search backwards to extract plan.

3 Knowledge Representation and Automated Reasoning

3.1 Intelligent Agents

An agent that enacts on its environment, based on **observations**, **prior knowledge**, **goals/preferences**, its **abilities** and sometimes **past experience**.



1. Find a representation for the problem.
2. Compute output using automatic reasoning.
3. Output mapped into solution to the problem.

Key Decisions

1. **Model of the environment**: can be expressed in terms of states / features / individuals and relationships between them.
2. **Uncertainty**: on state (partially observable state) / effects of actions.
3. **Preferences**: trade-off between desirability of various outcomes.
4. **Number of agents**: may be adversarial / cooperative.

3.2 Representation

Representation / Knowledge Schema Form of knowledge used in an agent.

Representation Internal representation / formalisation of knowledge. Should be:

- **Rich enough to express knowledge** needed to solve the problem.
- **Close to the problem**: declarative, compact and easy to maintain.
- **Amenable to efficient computation**: able to trade of accuracy and computation time.
- **Learnable**: can be automatically acquired from people, past experience, data.

Knowledge Base Representation of all the knowledge that is stored in an agent.

From Problem to Representation

- What level of abstraction?
  - High-level description is harder for a machine to comprehend. May abstract away important details.
  - Low-level description can be more accurate and predictive. But more difficult to reason with — more steps / actions to choose from.
  - Multiple levels of abstraction can solve these problems.

Solutions

- **Optimal**: best solutions according to some criteria.
- **Satisficing**: good enough according to some notion of an adequate solution.
- **Approximately optimal**: close enough to the optimal solution.
- **Probable**: likely to be a solution.

3.3 Reasoning

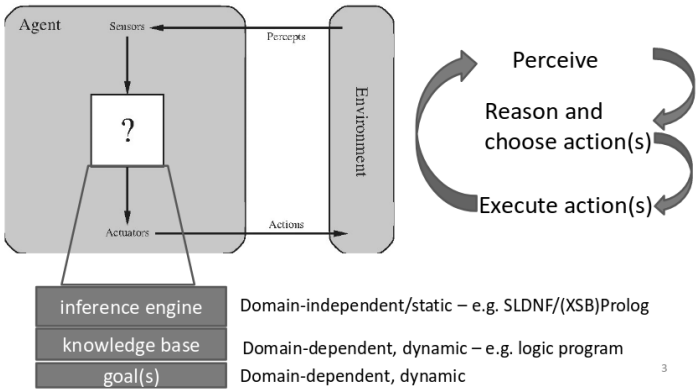
Reasoning Process made by a computational agent when searching for ways to determine how to complete its task.

- **Offline**: background knowledge needed to solve task is precomputed.
- **Online**: agents sense the environment to collect and use this data along with background knowledge, to decide at execution time what action to take.

Three Forms of Knowledge Inference

1. **Deductive**: reasoning from the general to reach the particular (what necessarily follows from the given?).
2. **Inductive**: reasoning from the specifics to reach the general (how can we generalise observations?).
3. **Abductive**: reasoning from observations to explanations (what causal relationships are there between knowledge and observations?).

3.4 Applying KRR



Ontologies Explicit, formal specification of a conceptualisation.

- **Concepts of the domain** e.g. classes such as professors, students, courses.
- **Relationships between these terms** e.g. all professors are staff members.

Allow for a shared understanding of a domain (**semantic interoperability**).

Resource Description Framework (RDF) Statements of the form object-attribute-value which assert properties of objects.

- **Triples**: (a, P, b) e.g. (introAI, isTaughtBy, ft).
- **Classes**: type(a, C).
- **Class hierarchies**: subClassOf(C, D).
- **Property hierarchies**: subPropertyOf(P, Q).

4 Deductive Reasoning

Clausal Form Resolution works with sets / conjunctions of clauses:

$$\neg p_1 \vee \cdots \vee \neg p_m \vee q_1 \vee \cdots \vee q_n$$

where each p_i and q_j is an atom.

- If $m = n = 0$: called the empty clause ($\square$).
- Every clause can be written as an implication $p_1 \wedge \cdots \wedge p_m \rightarrow q_1 \vee \cdots \vee q_n$.
- Every propositional logic formula can be written as a conjunction of clauses (conjunctive normal form).
- Every first-order logic sentence can be written as a conjunction of clauses (universal qualification + conjunctive normal form + Skolemisation).

Reaching Clausal Form

1. Change implication to disjunction: $p \rightarrow q$ goes to $\neg p \vee q$.
2. Push negation inwards using: $\neg (p \vee q) \equiv \neg p \wedge \neg q$ and $\neg (p \wedge q) \equiv \neg p \vee \neg q$.
3. (First-order) Push negation inwards using $\neg \exists X p(X) \equiv \forall X \neg p(X)$ and $\neg \forall X p(X) \equiv \exists X \neg p(X)$.
4. (First-order) Remove existential quantifier in $\exists Y F[Y]$ by:
   - If the subformula is not in the scope of any universal quantifier, replace Y with a new Skolem constant: e.g. $\exists Y p(Y) \implies p(c)$.
   - If the subformula occurs in the scope of universal quantifiers for $X_1, X_2, \ldots, X_n$, replace Y with a new Skolem term $sk(X_1, X_2, \ldots X_n)$: e.g. $\forall X \exists Y p(X, Y) \implies p(X, f(X))$.
5. (First-order) Move all universal quantifiers to the front of the sentence using: $\forall X \varphi_1(X) \wedge \varphi_2$ is equivalent to $\forall X (\varphi_1(X) \wedge \varphi_2)$ after appropriately renaming variables if X occurs in φ_2.
6. Distribute $\vee$ over $\wedge$ using: $p \vee (q \wedge s) \equiv (p \vee q) \wedge (p \vee s)$.

4.1 Unification

Universal Instantiation Combines two clauses:

$$\frac{\forall v \alpha}{\alpha\{v/g\}} \qquad \text{(for Alessandra: } \{v = g\}\text{)}$$

for any variable v and term g.

- $\sigma = \{v/g\}$ is a **substitution**. $\alpha\sigma$ means the formula obtained from α by applying σ.
- a' is an **instance** of a if $a = a\sigma$ for some σ.

Unification A substitution σ unifies atomic sentences p and q if $p\sigma = q\sigma$.

- E.g. $p = \text{Knows}(\text{John}, x)$, $\text{Knows}(y, \text{OJ})$, $\sigma = \{x/\text{OJ}, y/\text{John}\}$.

Most General Unifier θ is a most general unifier of formulas α and β if:

1. θ unifies formulas α and β.
2. If σ is any other unifier of α and β, then $\alpha\sigma$ is an instance of $\alpha\theta$ (i.e. $\alpha\sigma = (\alpha\theta)\sigma'$ for some σ').

4.2 Resolution

4.2.1 Propositional Logic

Resolution Inference Rule Combines two clauses:

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Applied repeatedly until $\square$ is derived.

Soundness of Resolution If $S \vdash P$ by resolution, then $S \vDash P$.

Refutation Completeness of Resolution

- If conjunction of propositional clauses is unjustifiable, we will end up with $\square$.
- To prove a goal P is entailed by a set of sentences S ($S \vDash P$):
  1. Compute CNF of S and $\neg P$.
  2. Apply resolution to results of step 1 to obtain $\square$.

4.2.2 First-Order Logic

Resolution Inference Rule For proof by resolution, draw a tree.

$$\frac{\alpha \vee \beta, \neg\beta' \vee \gamma}{(\alpha \vee \gamma)\,\theta} \text{ where } \theta \text{ is a mgu of } \beta \text{ and } \beta'$$

after appropriately renaming variables to be distinct. Full version:

$$\frac{\alpha^1 \vee \cdots \vee a^{j-1} \vee a^j \vee a^{j+1} \vee \cdots \vee \alpha^m, \beta^1 \vee \cdots \vee \beta^{k-1} \vee \beta^k \vee \beta^{k+1} \vee \cdots \vee \beta^n}{(\alpha^1 \vee \cdots \vee a^{j-1} \vee a^{j+1} \vee \cdots \vee \alpha^m, \beta^1 \vee \cdots \vee \beta^{k-1} \vee \beta^{k+1} \vee \cdots \vee \beta^n)\,\theta}$$

where θ is a mgu of α^j and $\neg\beta^k$.

4.2.3 Generalised Modus Ponens

$$\frac{p'_1, p'_2, \ldots, p'_n, (q \leftarrow p_1, p_2, \ldots, p_n)}{q\theta}$$

where $p'_i\theta = p_i\theta$.

4.3 SLD Resolution

Definite Clauses Clauses of the form (for atoms $p, q_1, \ldots, q_n$)

$$p \leftarrow q_1, q_2, \ldots, q_n$$

- p is the **head** and $q_1, \ldots, q_n$ the **body** of the clause.
- If $n = 0$, p is a **fact**.
- A set of definite clauses is a **logic program** / **knowledge base**.
- A **Horn clause** / **denial** is a definite clause or $\leftarrow q_1, \ldots, q_n$.

SLD Resolution Linear Resolution for Definite clauses with Selection function

$$\frac{\overbrace{\neg\alpha^1 \vee \cdots \vee \neg\alpha^j \vee \cdots \vee \neg\alpha^m}^{\text{Goal (denial)}}, \overbrace{\alpha \vee \neg\beta^1 \vee \cdots \vee \neg\beta^n}^{\text{Definite Clause}}}{(\neg\alpha^1 \vee \cdots \vee \neg\beta^1 \vee \cdots \vee \neg\beta^n \vee \cdots \vee \neg a^m)\,\theta}$$

where θ is a mgu of α^j and α.

Alternative View SLD Resolution

$$\begin{aligned} &\leftarrow L_1, \ldots, L_{j-1}, B, L_{j+1}, \ldots, L_n \\ \theta_i \mid &\text{ match } B \text{ with } B' \leftarrow M_1, \ldots, M_k \text{ since } B\theta_i = B'\theta_i \\ &\leftarrow (L_1, \ldots, L_{j-1}, M_1, \ldots, M_k, L_{j+1}, \ldots, L_n)\,\theta_i \end{aligned}$$

- Answer is computed from composition of all mgus.
- Sub-goal B can be chosen in any way.
- There may be many choices for the matching clause. Forms a SLD tree.

4.4 Forward and Backward Chaining

To prove that $S \vDash P$:

1. Compute the CNF S' of S.
2. Compute the CNF NP' of $\neg P$.
3. If S' is a set of definite clauses (logic program) and NP' is a Horn clause $\leftarrow q_1, \ldots, q_n$, then apply GMP to derive $q_1, \ldots, q_n$ from S'.

Forward Chaining (Bottom-Up Computation)

1. Split S' into facts E and definite clauses (rules) Pr.
2. Apply the rules in Pr to the facts in E using GMP to derive implied facts E'.

3. Add E' to E.
4. Repeat until no new facts are generated.
5. Succeed iff $q_1, \ldots, q_n$ are in E.

Likely to produce a lot of irrelevant facts.

Backward Chaining (Top-Down / Goal-Directed Computation) Start with goals $q_1, \ldots, q_n$ and $\theta = \{\}$. To solve a goal G wrt θ:

1. If there is a matching fact G' in S', add mgu σ to θ ($G\sigma = G'\sigma$).
2. For each rule $G' \leftarrow G_1, \ldots, G_m$ in S' whose head G' matches G via mgu σ', solve goals $G_1\sigma', \ldots, G_m\sigma'$ wrt to $\theta + \{\sigma'\}$.
3. Repeat until there are no goals to solve, return θ.

4.5 Semantics of Definite Clauses

- **Herbrand universe**: set of all ground terms that can be constructed from constant and function symbols in S.
- Each set of definite clauses S can be can be equated to the set of all its ground instances over the underlying Herbrand universe.

Classical Models Interpretations of S:

- Mappings from ground terms of S to elements of some domain D.

Models of S:

- Interpretations of S in which every clause of S is true.

Herbrand Models Models where ground terms denote themselves (i.e. domain is Herbrand universe of S).

- S has a model iff S has a (minimal) Herbrand model.

Immediate Consequence Operator T_s

- **Herbrand Base** HB: Set of all ground atoms constructed from predicate symbols in S over the Herbrand universe of S.
- Given $X \subseteq HB$, $T_s(X) = \{a \in HB \mid a \leftarrow b_1, \ldots, b_m \in S, \{b_1, \ldots, b_m\} \subseteq X\}$.
- Least fixed point, given by $T_s \uparrow^{\omega} = \cup_{i=0}^{\infty} T_s(\emptyset)$, which is the minimal Herbrand model of S.

4.6 SLD Resolution is Complete

If $S \vDash P\sigma$ (i.e. $P\sigma$ belongs to the minimal Herbrand model of S), then there exists and SLD-refutation of P to obtain $\square$ with answer θ and a substitution of ξ such that $P\sigma = P\theta\xi$.

Example

$$
\begin{aligned}
&S = \{P(x,y) \leftarrow Q(x), Q(1) \leftarrow, R(2) \leftarrow\} \\
&\leftarrow P(x,y) \\
&\leftarrow Q(x) \\
&\square\theta = \{x/1\}
\end{aligned}
$$

$P(1,2)$ belongs to the minimal Herbrand model of $S = \{Q(1), R(2), P(1,1), P(1,2)\}$ with $\xi = \{y/2\}$.

5 Deductive Reasoning with NAF

5.1 Non-monotonic Reasoning

Non-monotonic Logics Conclusions can be invalidated by adding more knowledge.

Default Reasoning In the absence of information to the contrary, results follow by default.
E.g.

$$
\begin{aligned}
&\text{BIRDS} = \text{flies}(X) \leftarrow, \neg\text{penguin}(X), \neg\text{wounded}(X), \ldots \\
&\text{BIRDS} \cup \{\text{bird}(\text{tweety})\} \vDash^{*} \text{flies}(\text{tweety}) \\
&\text{BIRDS} \cup \{\text{bird}(\text{tweety})\} \cup \{\text{penguin}(\text{tweety})\} \nvDash^{*} \text{flies}(\text{tweety})
\end{aligned}
$$

Closed World Assumption If α is not present in some DB, then assume $\neg\alpha$.

Negation as Failure $\neg B$ succeeds when all attempts to prove B fail (finitely).

Credulous vs. Sceptical Reasoning Assume Quakers are pacifists and republicans are not. Nixon is a Quaker and a Republican. Is Nixon a pacifist or not?

- **Sceptical / cautious reasoning**: No.
- **Credulous / brave reasoning**: Yes, to both.

5.2 SLDNF

Derivation steps:

- Select any positive literal $L_j = B$ from the current goal:

$$\begin{gathered} \leftarrow L_1, \ldots, L_{j-1}, B, L_{j+1}, \ldots, L_n \\ \theta_i \mid \text{match } B \text{ with } B' \leftarrow M_1, \ldots, M_k \text{ since } B\theta_i = B'\theta_i \\ \leftarrow (L_1, \ldots, L_{j-1}, M_1, \ldots, M_k, L_{j+1}, \ldots, L_n)\,\theta_i \end{gathered}$$

- Select any (**safe** / **ground** - otherwise will **flounder**) negative literal $L_j = \text{not } B$ from the current goal:

$$\begin{gathered} \leftarrow L_1, \ldots, L_{j-1}, \text{not } B, L_{j+1}, \ldots, L_n \\ \theta_i = \{\} \mid \text{all ways of computing goal } B \text{ must fail (finitely)} \\ \leftarrow (L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_n)\,\theta_i \end{gathered}$$

Properties Consider

$$\begin{gathered} p(X) \leftarrow q(X), \text{not } r(X) \\ q(a) \\ r(b) \end{gathered}$$

- **Non-monotonic**: $S \vdash_{\text{SLDNF}} p(a)$ but $S \cup \{r(a)\} \nvdash_{\text{SLDNF}} p(a)$.
- Builds in a kind of **CWA**: $S \cup \{r(a)\} \vdash_{\text{SLDNF}} \neg p(a)$.

5.3 Semantics for Normal Logic Programs

5.3.1 (Clark) Completion Semantics

The **completion** of S is found by:

1. Rewrite all rules in S as rules of the form

$$p\left(\overrightarrow{X}\right) \leftarrow \overrightarrow{X} = \overrightarrow{t}, \text{body}$$

   so $p(X, 3) \leftarrow q(X), \text{not } r(3)$ becomes $p(X, Y) \leftarrow Y = 3 \wedge q(X) \wedge \neg r(3)$

2. Combine all rules with the same head so

$$p\left(\overrightarrow{X}\right) \leftarrow \overrightarrow{X} = \overrightarrow{t}_1, \text{body}_1, \ldots, p\left(\overrightarrow{X}\right) \leftarrow \overrightarrow{X} = \overrightarrow{t}_n, \text{body}_n$$

   becomes

$$p\left(\overrightarrow{X}\right) \leftrightarrow \overrightarrow{X} = \overrightarrow{t}_1, \text{body}_1 \vee \cdots \vee p\left(\overrightarrow{X}\right) \leftarrow \overrightarrow{X} = \overrightarrow{t}_n, \text{body}_n$$

3. If some $q(\_)$ is not the head of any rule in S, add $q\left(\overrightarrow{X}\right) \leftrightarrow \text{false}$.
4. Add Clark's Equality Theory:
   (a) $f\left(\overrightarrow{t}\right) \neq g\left(\overrightarrow{s}\right)$ for all pairs of distinct terms in the Herbrand universe of S.
   (b) $X = X, X = Y \rightarrow Y = X, X = Y \wedge Y = Z \rightarrow X = Z$.
   (c) $\overrightarrow{X} = \overrightarrow{Y} \rightarrow p\left(\overrightarrow{X}\right) = p\left(\overrightarrow{Y}\right)$ for all predicate symbols p in S.
   (d) $\overrightarrow{X} \neq t\left[\overrightarrow{X}\right]$ for all terms t where $\overrightarrow{X}$ occurs.

Soundness of SLDNF

- If there is a SLDNF-refutation of P from S with answer θ then Completion $(S) \models P\theta$.
- If there is a **finitely-failed** SLDNF-refutation of P from S then Completion $(S) \models \neg P$.

5.3.2 Well-Founded Model Semantics

(In, Out) such that:

1. $\text{In} \subseteq HB$.
2. $\text{Out} \subseteq HB^{\text{not}} = \{\text{not } a \mid a \in HB\}$

- Atoms in $HB - (\text{In} \cup \{a \mid \text{not } a \in \text{Out}\})$ are 'undecided'.
- For $S = \{p \leftarrow \text{not } p\}$, the well-founded model is $(\{\}, \{\})$ so p is undecided.

5.3.3 Stable Model Semantics

Given a normal logic program S and $X \subseteq HB$, the reduce of S by X, S^X is obtained by:

1. Eliminate all rules with not p in the body, for every $p \in X$.
2. Eliminate all negative literals from the body of all remaining rules.

$X \subseteq HB$ is a **stable model** of S iff the least Herbrand model of S^X is X.

- **Credulous reasoning**: whatever holds in some stable model.
- **Sceptical reasoning**: whatever holds in all stable models.

6 Abductive Reasoning

(KB, A, IC) is an **abductive logic program** where:

- KB is the theory (a set of normal clauses).
- A is the set of aducibles in BK (ground **undefined** literals).
- IC is a set of integrity constraints (denials - must be false).
- G is the goal / observation.

Properties of Explanations An **abductive solution** is a set Δ of ground literals that are:

1. **Basic**: should be restricted to **aducibles** (ground literals with predicates that are not defined in the theory).
2. **Minimal**: should not be subsumed by other explanations.
3. **Consistent**: with the given theory, including integrity constraints.

Formally:

1. $\Delta \subseteq A$: belongs to aducibles.
2. $KB \cup \Delta \models G$: provides missing information needed to solve goal.
3. $KB \cup \Delta \not\models \bot$: consistent with knowledge base.
4. $KB \cup \Delta \models IC$: satisfies integrity constraints.

6.1 Abduction by Extending SLDNF

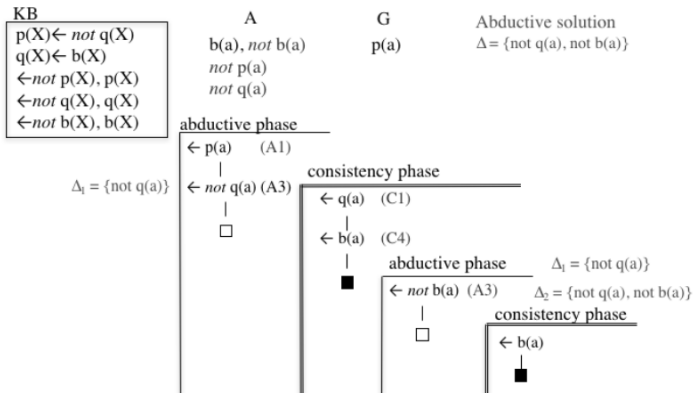
Abductive Derivation

- Perform SLDNF but maintain an accumulator Δ to which unproved sub-goals are added as assumptions.
- Order in which literals appear in a clause only influences order in which aducibles are added to explanation (not the explanation itself).

Consistency Derivation (Special Cases)

- **Integrity constraints**: making an assumption on aducibles in integrity constraints requires checking the satisfiability of these integrity constraints.
  - Resolve assumed literal with all relevant integrity constraints and ensure it fails.
- **Negation as failure**: requires adding a consistency phase: look for explanations, in terms of aducibles, for failure.
  - Implicitly add all negative literals to A.
  - Implicitly add $\leftarrow P, \text{not } P$ to IC for all predicates P.

6.2 Operational Semantics of Abduction



Abductive Phase Consider each literal in the goal, left to right. Start with an abductive phase.

1. **The current goal is not an aducible**. Initiate SLDNF derivation with current goal to be proved by refutation. If literal is positive, use normal resolution, otherwise go to case 3.
2. **The current goal is an aducible**. If the goal is already part of Δ, then the resolution is already proved and we proceed to the next goal.
3. **The current goal is an aducible not yet assumed**. Assume the literal. Trigger a consistency phase (failure derivation). If it does derive a failure, continue with the updated Δ.

Consistency Phase In the consistency phase, consider each literal from each constraint (denial) which includes the assumed literal.

1. **The literal is not an aducible**. Then apply a SLDNF failure derivation of this literal.
2. **The literal is already assumed in** Δ. This literal succeeds, so we need to check the rest to prove that it fails.
3. **The literal's negation is already assumed in** Δ. This literal fails, and so the current constraint is satisfied. Continue to the next constraint.
4. **Otherwise**. Trigger an abductive derivation of this literal's negation. If it succeeds, then this constraint is satisfied. Continue to the next constraint with the updated Δ.

6.3 Semantic Properties of Abduction

Knowledge Assimilation Explanation of new information computed abductively and added to KB.

Non-Monotonicity Use abduction for default reasoning. E.g. suppose $\text{fly}(X) \leftarrow \text{bird}(X)$ by default, but $\text{bird}(X) \leftarrow \text{penguin}(X)$ and $\neg\text{fly}(X) \leftarrow \text{penguin}(X)$:

$$KB \begin{cases} \text{bird}(X) \leftarrow \text{penguin}(X) \\ \text{not\_fly}(X) \leftarrow \text{penguin}(X) \\ \text{fly}(X) \leftarrow \text{bird}(X), \text{birdsFly}(X) \end{cases}$$
$$IC \leftarrow \text{birdsFly}(X), \text{not\_fly}(X)$$

Or using NAF:

$$KB \begin{cases} \text{bird}(X) \leftarrow \text{penguin}(X) \\ \text{abnormal}(X) \leftarrow \text{penguin}(X) \\ \text{fly}(X) \leftarrow \text{bird}(X), \text{not abnormal}(X) \end{cases}$$
$$IC\emptyset \ (\rightarrow \text{abnormal}(X), \text{not abnormal}(X) \text{ implicitly})$$

I.e. $KB \vdash Q$ under SLDNF if Q has an abductive solution after:

1. Renaming all uses of not $q(X)$ to $q*(X)$.
2. Adding $q*(X), q(X)$ to IC.

6.4 Applications of Abduction

- Diagnosis Problems
- Planning

Event Calculus Logical framework for representing and reasoning about events and effects over time.

- **Ontology** consists of events, fluents and time.
- **Signature** includes:
  - initiates(e, f, t): Fluent f starts to hold after event e at time t.
  - terminates(e, f, t): Fluent f stops holding after event e at time t.
  - initially(f): Fluent f holds at the beginning.
  - happens(e, t): Event e happens at time t.
  - holdsAt(f, t): Fluent f holds at time t.
  - clipped(t_1, f, t_2): Fluent f is terminated between times t_1 and t_2.

Description includes two theories:

- **Domain independent theory**: General rules for when fluents hold at certain times.
  - $\text{holdsAt}(f, t_2) \leftarrow \text{initially}(f), \text{not clipped}(0, f, t_2)$: initially true fluents hold until an event terminates them.
  - $\text{holdsAt}(f, t_2) \leftarrow \text{initiates}(e, f, t_1), \text{happens}(e, t_1), t1 < t2, \text{not clipped}(t_1, f, t_2)$: fluents initiated by an event hold until an event terminates them.
  - $\text{clipped}(t_1, f, t) \leftarrow \text{happens}(e, t_2), \text{terminates}(e, f, t_2), t_1 < t_2, t_2 < t$: fluents only change status via terminating events.
- **Domain dependent theory**: Particular effects of events or actions, using initiates(.) $\leftarrow$ BODY and terminates(.) $\leftarrow$ BODY, as well as initial state (using initially(.)) and event occurrences (using happens(.)).
  - Preconditions are captured in the conditions in initiates axioms.
  - Effects are expressed by head atoms of initiates and terminates rules.

Event Calculus Abductive Planning Given

- DI: domain independent theory.
- DD: domain dependent theory.
- A: set of ground aducibles.
- IC: integrity constraints (usually $\{\text{happens}(E_1, T), \text{happens}(E_2, T), E_1 \neq E_2\}$).
- G: goal state.

A plan is some $P = (As, TC)$ where

- $As = \{\text{happens}(e_i, t_i), \text{happens}(e_j, t_j), \dots\}$.
- $TC = \{t_i < t_j, \dots\}$.

such that

- $DI \cup DD \cup P \models G$.
- $DI \cup DD \cup P \cup IC$ is consistent.

Setting $KB = DI \cup DD$ and $\Delta = P$:

- $KB \cup \Delta \models G$.
- $KB \cup \Delta \cup IC \not\models \bot$.

7 SAT Solving

Given a propositional formula φ (usually in CNF), SAT on φ means to determine if there exists a variable assignment under which φ evaluates to true.

Why CNF?

- Efficient algorithms to transform formulae into CNF.
- Can exploit CNF: easier to detect conflicts.

SAT Algorithms

- Complete algorithms:
  - **Proof systems**: e.g. truth table / natural deduction.
  - **Davis-Putman** (DP): based on resolution.
  - **Stalmark's method**.
  - **Davis-Logemann-Loveland** (DLL/DPLL): search-based.
  - **Conflict-Driven Clause Learning** (CDCL).
- Incomplete algorithms:
  - Local search / hill climbing.
  - Genetic algorithms.

7.1 DP Algorithm

For a set S of clauses, for each variable in S:

1. For each clause C_i containing the variable and each clause C_j containing its negation, resolve C_i and C_j. (Fail if resolvent is $\square$).
2. Add the resolvent to S.
3. Remove S from all clauses.

Final set of variables informs a **partial assignment**.

Improvements

- Ignore when resolution generates a tautology (literal and its negation).
- **Pure literal rule**
  - **Pure literal**: only ever occurs with the same polarity.
  - Clauses containing pure literals can be removed.
  - Remember required assignments.
- **Unit propagation rule**
  - **Unit clause**: clause with a single literal.
  - Satisfy the literal by making it true.
  - Propagate the assignment by removing clauses that contain the literal, and removing the negated literal from any clauses that contain it.

7.2 DLL Algorithm

Key Idea Instead of eliminating variables, split on a variable at each step.

Simplifications Given a set Sc of clauses:

- UNSAT (Sc): if Sc contains $\{\}$ then formula is unsatisfiable.
- SAT (Sc): if $Sc = \{\}$ then formula is satisfiable.
- MULT (L, Sc): if literal occurs more than once in a clause in Sc, all but one can be deleted.
- SUB (C, Sc): A clause in Sc can be deleted if it is a superset of another clause in Sc.
- TAUT (C, Sc); A clause in Sc that contains a literal and its negation can be deleted.
- PURE (L, Sc): If L is a pure literal in Sc, delete all clauses containing L and add L to the assignment.
- UNIT (L, Sc): If Sc contains a unit clause $\{L\}$, delete all clauses including L, remove $\neg L$ from the remaining clauses, and add L to the assignment.
- SPLIT (L, Sc): If Sc contains clauses of the from $\{C_k, L\}$ and $\{C_m, \neg L\}$, branch the computation (adding L and $\neg L$ to each branch's assignment respectively), and regenerate Sc using the UNIT rule.

8 Answer Set Programming

8.1 Semantics

All rules must be safe:

- A rule R is safe, if every variable in R occurs in body$^+$ (R).

To check with X is an answer set of a program P:

1. Find the **relevant grounding** $\mathcal{RG}(P)$:

   (a) For each rule R in P, generate all rules which are ground instances R_g of R, s.t. for each atom A in $\text{body}^+(R_g)$ there is at least one rule already in $\mathcal{RG}(P)$ with A as the head.

2. Calculate the reduct of $\mathcal{RG}(P)$ w.r.t. X, $\mathcal{RG}(P)^X$:

   (a) Remove any rule whose body contains the negation of an atom in X.
   (b) Delete any negation from the remaining rules.

3. Check whether $X = M\left(\mathcal{RG}(P)^X\right)$ (the LHM of the reduct).

   (a) Find LHM by starting with $M = \{\}$ iteratively adding any atom that is the head of a rule in P whose body is already satisfied by M.

Relationship between ASP and Clark Completion The answer sets of P are the models of completion(P) which have no non-empty unfounded subsets w.r.t. P.

- Given an interpretation I, $U \subseteq I$ is an **unfounded subset** w.r.t. P if there's no rule R in P that proves anything in U without using elements of U. I.e. no rule s.t.:

  1. I satisfies $\text{body}(R)$.
  2. $\text{head}(R) \in U$.
  3. $I \cap \text{body}^+(R) = \emptyset$.

Brave and Cautious Semantics

- P **bravely** entails A if **some** answer set of P contains A.
- P **cautiously** entails A if **every** answer set of P contains A.

8.2 Constraints

Choice Rules If the body is satisfied by an answer set, then the number of h_is should be between `lb` and `ub`.

$$\texttt{lb}\,\{h_1; \ldots; h_m\}\,\texttt{ub} \;:-\; b_1, \ldots, b_n$$

Now when constructing the reduct:

- If X satisfies the head of R, then for each atom h that occurs both in the head and in X, P^X contains $h \;:-\; \text{body}^+(R)$.
- If X doesn't satisfy the head of R, then P^X contains $:-\; \text{body}^+(R)$.

Choice rules can generate non-minimal answer sets.

Disjunction Represented using ;. Unlike choice sets, all answer sets are minimal models.

Hard Constraints Eliminates any answer that satisfies $b_1 \wedge \cdots \wedge b_n$.

$$:- b_1, \ldots, b_n$$

Abduction in ASP Rewrite the aducibles as a choice rule, and the goal as a constraint.

8.3 Aggregates

$$\texttt{lb\#agg}\,\{e_1; \ldots; e_n\}\,\texttt{ub}$$

is satisfied by X iff

$$\texttt{lb\#agg}\,(\text{eval}\,(X, \{e_1, \ldots, e_n\}))\,\texttt{ub}$$

- Variables which only occur inside aggregates are **local**, and produce multiple aggregate elements, instead of multiple rules.
  - **Safety**: a rule with an aggregate is unsafe if a local variable occurs in an aggregate element but doesn't occur in a positive literal on the RHS of the element.
- Semantics usually defined by FLP reduct.

$$P^X_{FLP} = \{r \in P \mid X \vDash \text{body}(I)\}$$

  but this doesn't always have a minimal model.

8.4 Optimisation

Weak Constraints

$$:\sim\; b_1, \ldots, b_m.\,[\texttt{wt@lev}, t_1, \ldots, t_n]$$

The score at a priority level p is

$$\text{score}\,(P, p, X) = \sum \texttt{wt}.\,[\texttt{wt@p}, t_1, \ldots, t_n]_{\in \text{Weak}(P,X)}$$

- X_1 is preferred to X_2 if for the highest priority where their scores differ, X_1 has a lower score.