Artificial Intelligence: Set-based Search

CPSC 433: Artificial Intelligence
Fall 2022

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Thursday, September 29, 2022

UNIVERSITY OF CALGARY

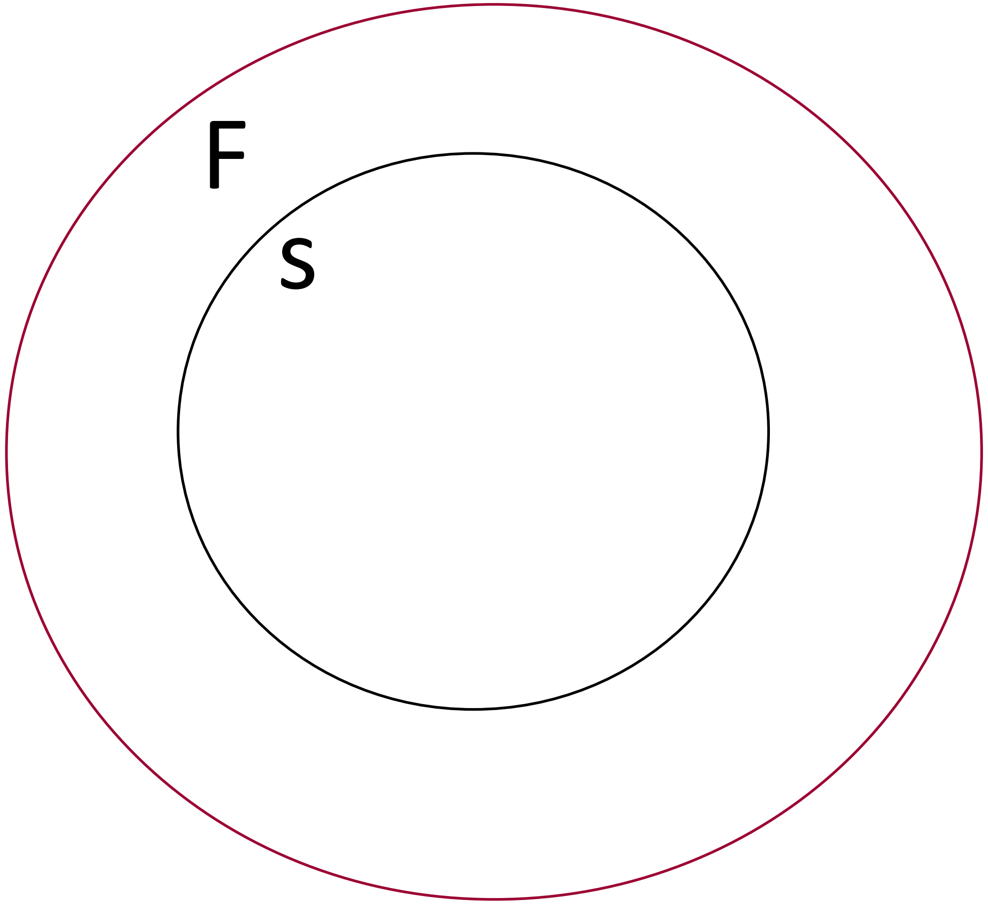
Set-based Search?

Basic Idea:

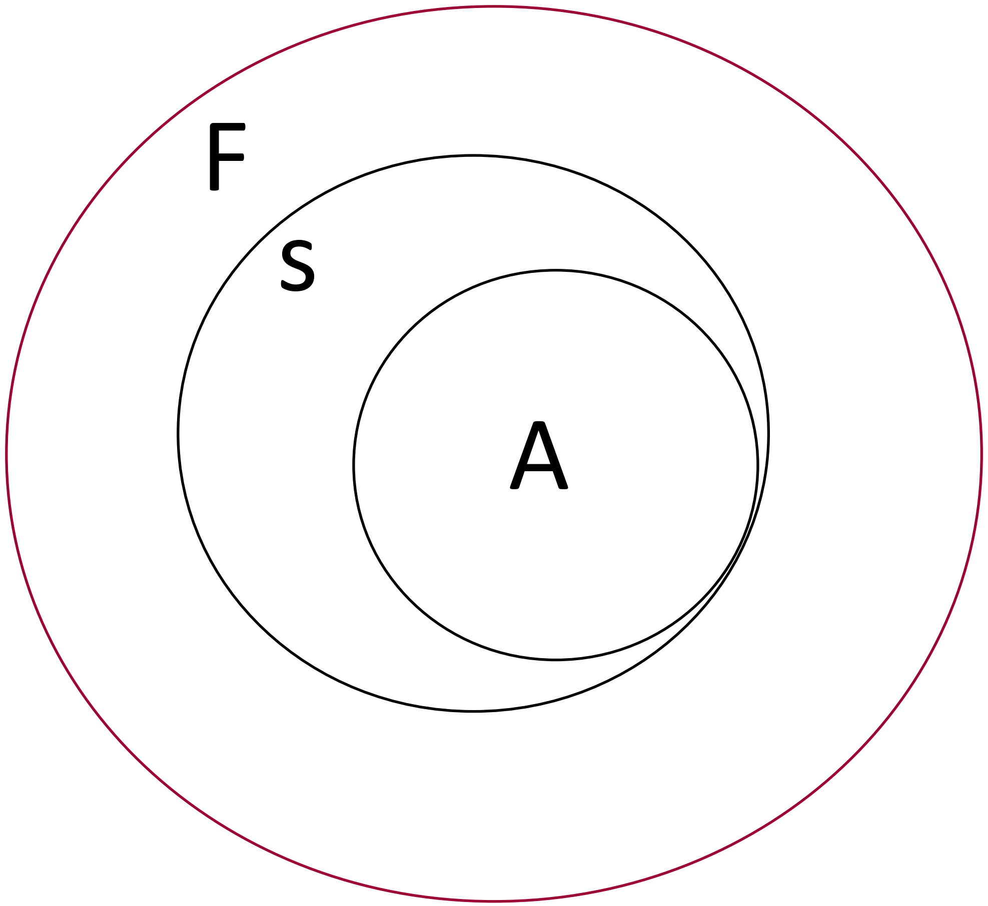
1. We have a collection of pieces of information (facts) that is (mostly) growing during the performance of a search
   - a relation between the different pieces is either not known, not of interest or describing only consequences of facts.

☞Represent collection as a set, go from one set to successor by adding/deleting facts according to rules
   - taking into account other facts already in the collection.

Venn Diagram of Facts and States

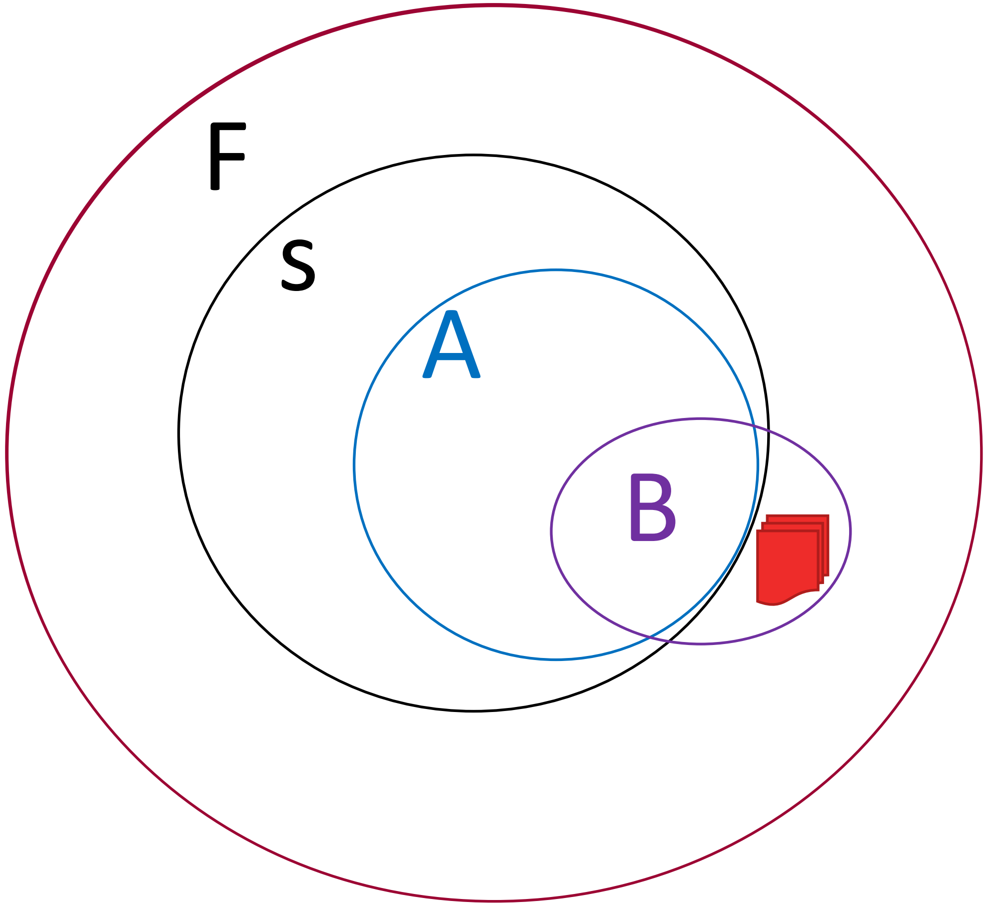


Venn Diagram of Facts and States



UNIVERSITY OF CALGARY

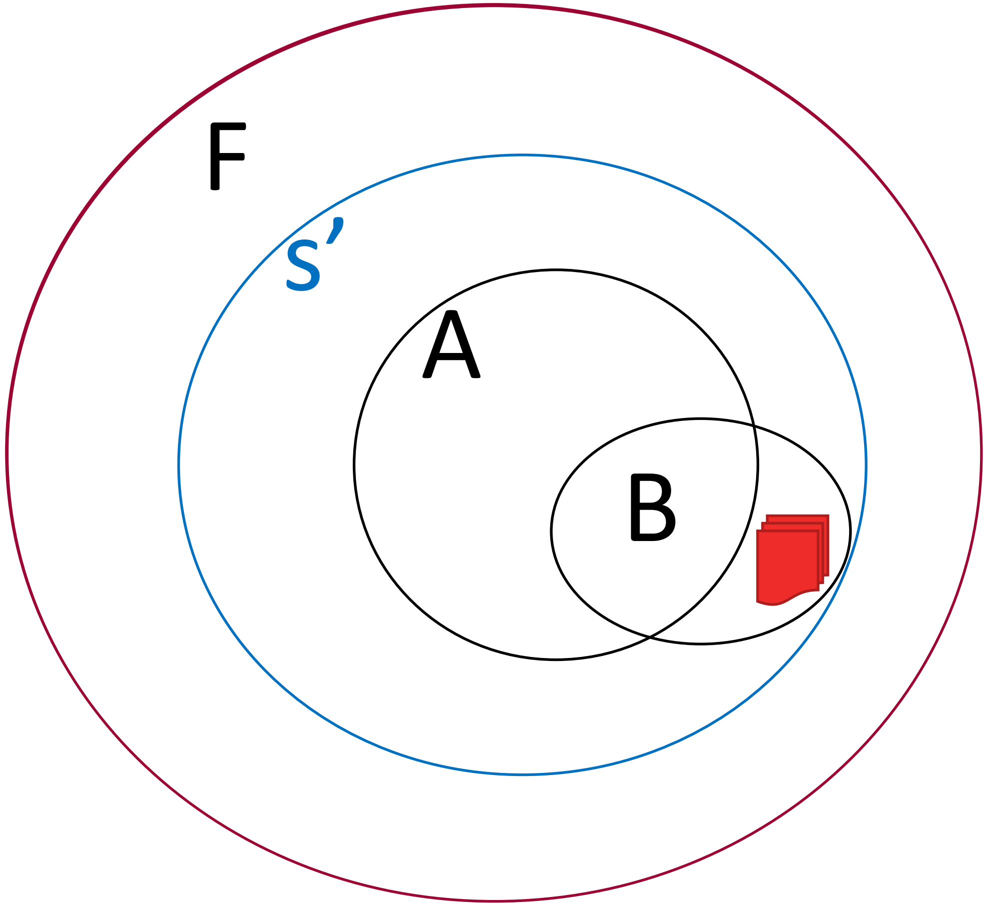
Venn Diagram of Facts and States



$A \rightarrow B$
B contains new facts!
B may or may not drop facts from A.

UNIVERSITY OF CALGARY

Venn Diagram of Facts and States



New state s' includes B (would drop any facts B dropped as well)

Definitions

Formal Definitions: Model

Set-based Search Model $A_{set} = (S_{set}, T_{set})$

F set of facts

$Ext \subseteq \{A \rightarrow B \mid A, B \subseteq F\}$ extension rules i.e. rules where one set of facts A lets me create another set of facts B

$S_{set} \subseteq 2^F$ set of possible states, is subset of the power set of Facts

$T_{set} \subseteq S_{set} \times S_{set}$ transitions between states, but more specifically

$T_{set} = \{(s, s') \mid \exists A \rightarrow B \in Ext$ with $A \subseteq s$ and $s' = (s - A) \cup B\}$

Transitions exists where we use extension rule to go from state with facts in A to facts in B

Less formally: Model

- F can consist of solution pieces, solution candidates, parts of a world description, etc.
- With Ext we try to get more solution pieces, better candidates, more explicit parts of the description
- Or we eliminate wrong pieces, less good solutions, unnecessary explicit parts
- We construct the new parts using parts we already have

☞We make implicit knowledge explicit

Formal Definitions: Search Process

Set-based Search Process $P_{set} = (A_{set}, Env, K_{set})$

K_{set}: $S_{set} \times Env \rightarrow S_{set}$ search control is a function K transitioning from current state to next state

$K_{set}(s,e) = (s - A) \cup B$ where

1. $A \rightarrow B \in Ext$
2. $A \subseteq s$
3. $\forall A' \rightarrow B' \in Ext$ with $A' \subseteq s$ holds: $f_{wert}(A,B,e) \leq f_{Wert}(A',B',e)$ [we selected a best rule (given in 1.) based on minimizing function f_{wert}]
4. $A \rightarrow B = f_{select}(\{A' \rightarrow B' | f_{wert}(A',B',e) \leq f_{wert}(A'',B'',e) \quad \forall A'' \rightarrow B'' \in Ext$ with $A'' \subseteq s\}, e)$ [TBD tie break that produces 1 rule out of many]

Formal Definitions: Search Process

Set-based Search Process $P_{set} = (A_{set}, Env, K_{set})$

$K_{set}:\ S_{set} \times Env \to S_{set}$ search control is a function K transitioning from current state to next state

$K_{set}(s,e) = (s - A) \cup B$ where

- Set-based search selects transition (extension rule change) based on f_{wert} and tie breaks with f_{select} if there were more than one.
- $f_{wert}: 2^F \times 2^F \times Env \to \mathbb{N}$ values each choice to a number
- $f_{select}: 2^F \times 2^F \times Env \to 2^F \times 2^F$ if more than one, picks one, could be random!

Less formally: Search Process

- The control selects the extension to apply by
  - Evaluating each applicable extension into a number (done by $\mathrm{f}_{\mathrm{wert}}$)
  - Considering only extensions with minimal evaluation
  - Use $\mathrm{f}_{\mathrm{select}}$ as tiebreaker
- Obviously, there usually are many different $\mathrm{f}_{\mathrm{wert}}$ and $\mathrm{f}_{\mathrm{select}}$ functions
- Sometimes $\mathrm{f}_{\mathrm{wert}}$ can also produce integers or real numbers

Formal Definitions: Search Instance

Search Instance $Ins_{set} = (s_0, G_{set})$

$s_0, s_{goal} \in 2^F$

$G_{set}: S \rightarrow \{yes, no\}$ goal condition (function on current state that halts)

$G_{set}(s_i) = yes$ if and only if $s_{goal} \subseteq s_i$ or there is no extension rule applicable in s_i

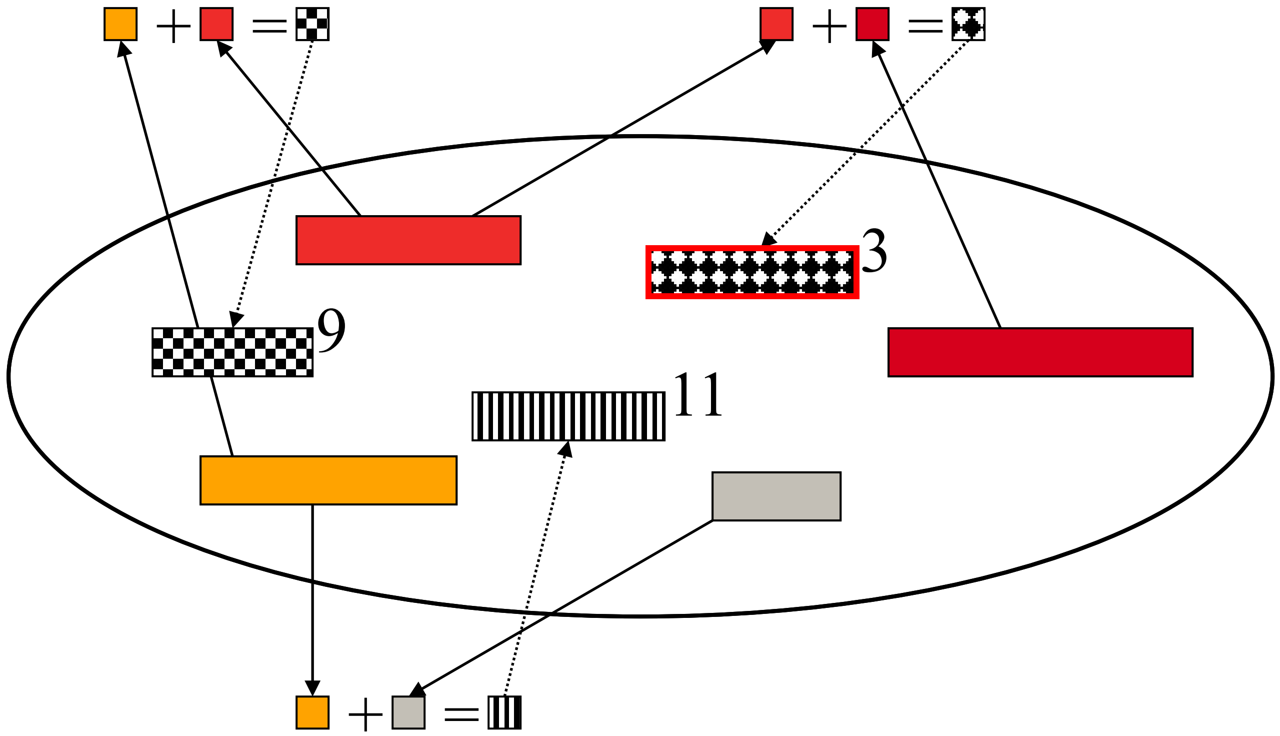
Less formally: Search Instance

- We start with the given solution pieces, some random solutions, or the given parts of the description (or …)
- We stop, if
  - a complete solution s_{goal} is part of the actual state or
  - a good enough candidate that is really a solution is found or
  - the description is good enough or
  - a time limit is reached

  i.e. if enough knowledge (s_{goal}) is made explicit

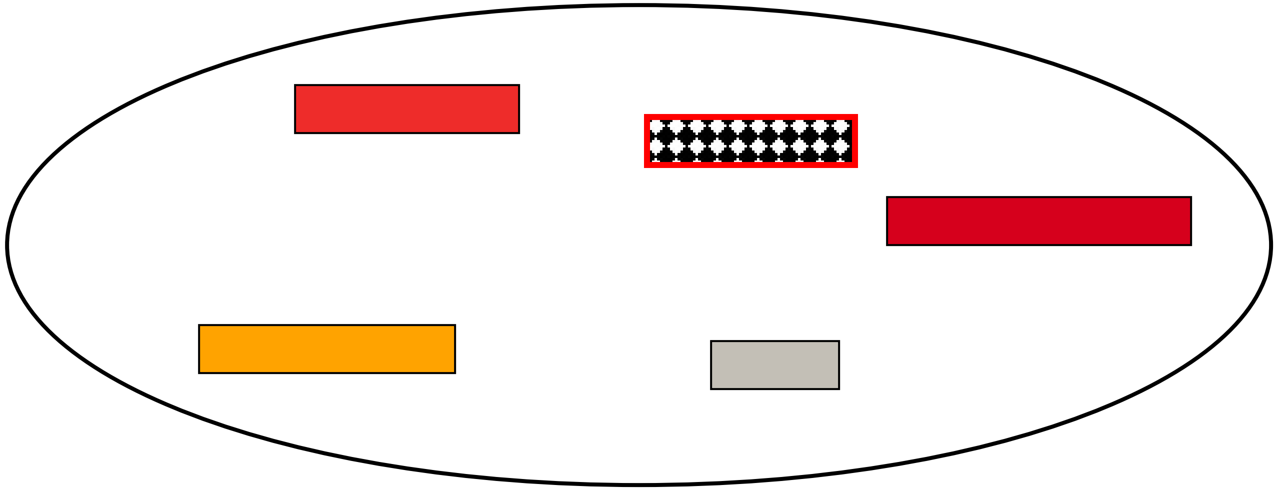
Visualize

Conceptual Example (I): Set-based Search



Conceptual Example (I): Set-based Search

Next state:



Design

Designing set-based search models

1. Identify set of facts F
2. Identify how you create new facts out of known facts (make sure that what you create are really facts!)
   ☞ Ext
3. You have your sets F and Ext that, with our definition earlier, are sufficient to define a set-based search model

Designing set-based search processes

1. Identify possible functions that measure a fact
2. Decide if it is not too computationally expensive to compute the right side of applicable rules
3. If it is not too expensive, define f_{wert} by measuring A and B using 1.
4. If it is too expensive, define f_{wert} by measuring only A using 1.
5. If you want to rely on random decisions (or include them), set f_{wert} constant
6. Identify rules that have the same f_{wert} -value and design f_{select} as tiebreaker (random decisions are best expressed using f_{select})

Review

Review of Logic Definitions

- Propositional logic – zeroth order logic, does not have predicates, just formulas of singular propositional symbols, often p,q,r,… combined with (or ∨, and ∧, not ¬, implication →, biconditional ↔) Ex. $\neg p \vee q \rightarrow r$
- First-order logic – formulas use variables, constants, predicates, functions, quantifiers there is ∃ and for all ∀, equality
- Variable – generally w,x,y,z
- Constant – generally a,b,c,d,…. Or sometimes alice, bob, carol, etc. or similar. Can replace a variable
- Predicate – a property or relation, P(a) would mean a constant a has property P, while P(x) would mean the same for indeterminate variable, returns truth value
- Function – constants are a subset of these with no parameters, generally f,g,h,etc. maps within domain of variables, f(x) -> y where both x,y are in domain of problem

Review of Logic Definitions

- Clause – a single logical formula
- Disjunction – or   Conjunction-and
- Conjunctive Normal Form (CNF) – a set of clauses changed to a form where it becomes a conjunction of clauses where each clause is a disjunction of literals
  - Have clauses A, B, C then conjunction of them becomes A and B and C
  - Every formula can be written in this form. Note negations and brackets are transformed by logical rules such that negations apply to predicates and brackets are around clauses
  - $\neg(B \vee C)$ becomes $(\neg B) \wedge (\neg C)$ or $(\mathrm{A} \wedge B) \vee C$ becomes $(A \vee C) \wedge (B \vee C)$

Review of Logic Definitions

- Unification – in our case used to attempt to find the most general unifier, which is a valid mapping of variable/constant/function mapping to make two terms the same, Ex. if I have f(a) and f(x) mapping x->a makes f(a)=f(a)
- Resolution – theorem proving technique, general process is to
  1. Take known clauses and negate the conclusion trying to be proven
  2. Then turn this into CNF
  3. Attempt to derive empty clause
  4. If found this indicates the set of clauses was not satisfiable
  5. This then means that the original conclusion was supported by the clauses

Review: Quick Resolution Example

- $433Inst(x) \rightarrow Cool(x), 433Inst(Jon)$ is $Cool(Jon)$?
- $433Inst(x) \rightarrow Cool(x) \land 433Inst(Jon) \land \neg Cool(Jon)$ set of clauses
- $(\neg 433Inst(x) \lor Cool(x)) \land (433Inst(Jon)) \land (\neg Cool(Jon))$ in CNF
- $(\neg 433Inst(x) \lor Cool(x)) \land (433Inst(Jon))$ resolve to $Cool(Jon)$
- $Cool(Jon) \land (\neg Cool(Jon))$ resolve to $\blacksquare$
- Therefore, the CNF form was unsatisfiable which means the original clauses agree with $Cool(Jon)$

Applied to Resolution

Concrete Example: Resolution (I)

- We describe our world by a collection of special logical formulas, so-called clauses:

$$\mathrm{L}_1\left(t_{1,1}, \dots, t_{1,n1}\right) \vee \cdots \vee L_m\left(t_{m,1}, \dots, t_{m,nm}\right)$$

where L_i predicate symbol or its negation, $t_{i,j}$ terms out of function symbols and variables (x,y…) variables in different clauses are disjunct

- Examples: $p \vee \neg q, P(a,b,x) \vee R(x,y,c), Q\big(f(a,b), g(x,y)\big), \neg Q(a,b)$
- A consequence we want to prove is negated, transformed into clauses and these clauses are added to the world.
- The consequence is proven, if the empty clause (■) can be deduced.

Concrete Example: Resolution (II)

- We derive new clauses by either Resolution or Factorization

Resolution:

$$\frac{C \vee P\,, D \vee \neg P'}{\sigma(C \vee D)} \qquad \text{if } \sigma = \text{mgu}(P, P')$$

mgu = most general unifier

Factorization:

$$\frac{C \vee P \vee P'}{\sigma(C \vee P)} \qquad \text{if } \sigma = \text{mgu}(P, P')$$

Concrete Example: Resolution: Unification (I)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $\mathrm{u} \approx v$, with $\perp$ (symbol for False) indicating failure

Extension rules:

Delete:

$$\frac{E \cup \{t \approx t\}}{E}$$

No longer need to maintain a unifier of something to itself

Concrete Example: Resolution: Unification (II)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $\mathrm{u} \approx v$, with $\bot$ indicating failure

Extension rules:

Decompose:

$$\frac{E\ \cup \{f(t_1, \dots, t_n) \approx f(s_1, \dots, s_n)\}}{E \cup \{t_1 \approx s_1, \dots t_n \approx s_n\}}$$

If you have function unified to same name function, can recompose unifier to only be unifying the internals

Concrete Example: Resolution: Unification (III)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $\mathrm{u} \approx v$, with $\perp$ indicating failure

Extension rules:

Orient:

$$\frac{E \cup \{t \approx x\}}{E \cup \{x \approx t\}}$$

t is not variable

Order of unifier can be changed

Concrete Example: Resolution: Unification (IV)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $u \approx v$, with $\perp$ indicating failure

Extension rules:

Substitute:

$$\frac{E \cup \{x \approx t, t' \approx s'\}}{E \cup \{x \approx t, t'[x \leftarrow t] \approx s'[x \leftarrow t]\}}$$

Can modify one unifier with another as long as x not in t

Concrete Example: Resolution: Unification (V)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $u \approx v$, with $\perp$ indicating failure

Extension rules:

Occurs check:

$$\frac{E \ \cup \{x \approx t\}}{\perp}$$

If x is in t we cannot unify them (think infinite expansion as issue)

Concrete Example: Resolution: Unification (VI)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $u \approx v$, with $\bot$ indicating failure

Extension rules:

Clash:

$$\frac{E \cup \{f(t_1, \ldots, t_n) \approx g(s_1, \ldots, s_n)\}}{\bot}$$

If $f \neq g$ we cannot unify them

Concrete Example: Resolution: Unification (VII)

Needed: Unification to compute mgu

Yet another set-based search problem:

States: set of term equations $u \approx v$, with $\perp$ indicating failure

Extension rules:

Delete, Decompose, Orient, Substitute, Occurs check, Clash

Goal condition: all equations in the state have form $x \approx t$ and Occurcheck and Substitute are not applicable

Unification/Resolution: Examples

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Unification:

(1) $f(g(x,y),c) \approx f(g(f(d,x),z),c)$

(2) $h(c,d,g(x,y)) \approx h(z,d,g(g(a,y),z))$

Examples for Resolution:

(1) $p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$

(2) $P(x) \lor R(x), \neg R(f(a,b)), \neg P(g(a,b))$

(3) $P(x) \lor R(y), \neg R(f(a,b)), \neg P(g(a,b))$

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Unification:

$\{\boldsymbol{f}(g(x,y),c) \approx \boldsymbol{f}(g(f(d,x),z),c)\}$ decompose

$\{g(x,y) \approx g(f(d,x),z),\ \boldsymbol{c} \approx \boldsymbol{c}\}$ delete

$\{\boldsymbol{g}(x,y) \approx \boldsymbol{g}(f(d,x),z)\}$ decompose

$\{\boldsymbol{x} \approx \boldsymbol{f}(\boldsymbol{d},\boldsymbol{x}),\ y \approx z\}$ occurs check $\perp$

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Unification:

$\{\boldsymbol{h}(c,d,g(x,y) \approx \boldsymbol{h}(z,d,g(g(a,y),z))\}$ decompose

$\{c \approx z, \boldsymbol{d} \approx \boldsymbol{d}, g(x,y) \approx g(g(a,y),z)\}$ delete

$\{\boldsymbol{c} \approx \boldsymbol{z}, g(x,y) \approx g(g(a,y),z)\}$ orient

$\{z \approx c, \boldsymbol{g}(x,y) \approx \boldsymbol{g}(g(a,y),z)\}$ decompose

$\{z \approx c, x \approx g(a,y), \boldsymbol{y} \approx \boldsymbol{z}\}$ substitute

$\{z \approx c, \boldsymbol{x} \approx \boldsymbol{g}(\boldsymbol{a},\boldsymbol{y}), y \approx c\}$ substitute

$\{z \approx c, x \approx g(a,c), y \approx c\}$ done

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Unification:

(1) $f(g(x,y),c) \approx f(g(f(d,x),z),c)$ occur check $\perp$

(2) $h(c,d,g(x,y)) \approx h(z,d,g(g(a,y),z))$ mgu = $\{z \approx c, x \approx g(a,c), y \approx c\}$

Examples for Resolution:

(1) $p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$

(2) $P(x) \lor R(x), \neg R(f(a,b)), \neg P(g(a,b))$

(3) $P(x) \lor R(y), \neg R(f(a,b)), \neg P(g(a,b))$

Concrete Example: Resolution (III)

(1) $p \vee q$

(2) $p \vee \neg q$

(3) $\neg p \vee q$

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

(5) $p \vee p$ resolve (1) and (2)

(6) p factorize (5)

(7) $\neg p \vee \neg p$ resolve (3) and (4)

(8) $\neg p$ factorize (7)

(9) $\blacksquare$ resolving (6) and (8)

Resolution:

$$\frac{C \vee P\,,\, D \vee \neg P'}{\sigma(C \vee D)}$$

Factorization:

$$\frac{C \vee P \vee P'}{\sigma(C \vee P)}$$

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Resolution:

(1) $P(x) \vee R(x)$

(2) $\neg R(f(a,b))$

(3) $\neg P(g(a,b))$

(4) $P(f(a,b))$ resolving (1) and (2) with $mgu = \{x \approx f(a,b)\}$

(5) $R(g(a,b))$ resolving (1) and (3) with $mgu = \{x \approx g(a,b)\}$

Can’t reach empty clause

Resolution:

$$\frac{C \vee P\,,\, D \vee \neg P'}{\sigma(C \vee D)}$$

Factorization:

$$\frac{C \vee P \vee P'}{\sigma(C \vee P)}$$

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Resolution:

(1) $P(x) \lor R(y)$

(2) $\neg R(f(a,b))$

(3) $\neg P(g(a,b))$

(4) $P(x)$ resolving (1) and (2) with $mgu = \{y \approx f(a,b)\}$

(5) ■ resolving (3) and (4) with $mgu = \{x \approx g(a,b)\}$

Resolution:

$$\frac{C \lor P, D \lor \neg P'}{\sigma(C \lor D)}$$

Factorization:

$$\frac{C \lor P \lor P'}{\sigma(C \lor P)}$$

UNIVERSITY OF CALGARY

Concrete Example: Resolution (III)

x,y,z are variables, rest are literals, functions, and predicates

Examples for Unification:

(1) $f(g(x,y),c) \approx f(g(f(d,x),z),c)$ occur check $\perp$

(2) $h(c,d,g(x,y)) \approx h(z,d,g(g(a,y),z))$ mgu = $\{z \approx c, x \approx g(a,c), y \approx c\}$

Examples for Resolution:

(1) $p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$ produced empty clause

(2) $P(x) \lor R(x), \neg R(f(a,b)), \neg P(g(a,b))$ couldn't reach empty clause

(3) $P(x) \lor R(y), \neg R(f(a,b)), \neg P(g(a,b))$ produced empty clause

Unification/Resolution: Set-Based

Concrete Example: Resolution (V)

Tasks:

- Describe Resolution as set-based search model
  - F, Ext
- Given the following control idea, describe formally a search control for your model, so that we have a search process:
  - f_{wert}, f_{select}

Perform factorization whenever possible; choose the smallest possible clauses for resolution; if several clause pairs are smallest, use an ordering $<_{Lit}$ on the predicates and terms

Concrete Example: Resolution (VI) Model

$$F = \{f_1, \dots, f_t\}$$

Concrete Example: Resolution (VI) Model

$$F = \{f_1, \ldots, f_t \mid f_i = \mathrm{L}_1(t_{1,1}, \ldots, t_{1,n1}) \vee \cdots \vee L_m(t_{m,1}, \ldots, t_{m,nm})\}$$

set of t facts where each fact is formed where L_i predicate symbol or its negation, $t_{i,j}$ terms out of function symbols and variables (x,y…) variables in different clauses are disjunct}

Concrete Example: Resolution (VI) Model

$$Ext = \{A \rightarrow B \mid A, B \subseteq 2^F \ and \ \}$$

Concrete Example: Resolution (VI) Model

$$Ext = \{A \rightarrow B \mid A, B \subseteq 2^F \text{ and } Resolution(A, B), Factorization(A, B)\}$$

Concrete Example: Resolution (VI) Model

$$Ext = \{A \rightarrow B \mid A, B \subseteq 2^F \ and \ (Resolution(A, B) \ or \ Factorization(A, B))\}$$

$$Resolution(A, B) = \frac{C}{D} where \ A = C \ and \ B = C \cup D$$

$$Factorization(A, B) = \frac{C}{D} where \ A = C \ and \ B = C \cup D$$

Resolution:

$$\frac{C \vee P \, , D \vee \neg P'}{\sigma(C \vee D)}$$

Factorization:

$$\frac{C \vee P \vee P'}{\sigma(C \vee P)}$$

Concrete Example: Resolution (VI) Process

$$f_{wert}(A, B, e) = \mathbb{N}$$

Concrete Example: Resolution (VI) Process

- $f_{wert}(A, B, e) = \mathbb{N}$
  - If $A \to B$ exists that fulfils $Factorization(A, B) = \frac{C}{D}$ with $D \notin s$ then $f_{wert}(A, B, e) = 0$ (always choose factorization)
  - if $A \to B$ exists that fulfills $Resolution(A, B) = \frac{C}{D}$ with $D \notin s$ then $f_{wert}(A, B, e) = size(A)$ where $size(A)$ is a summation of size of clauses in A (next do Resolution based on size)
- $f_{select}(\{A' \to B'\}, e) = A \to B$
  - where $A \to B$ is at index 0 after creating a sorted order of $\{A' \to B'\}$ according to ordering $<_{\text{Lit}}$ (use ordering for tie break) [there should exists no two clauses which cannot be ordered by $<_{\text{Lit}}$ as there are no duplicates]

Unification/Resolution: Set-Based: Applied

Concrete Example: Resolution (VI)

Tasks (cont.):

- Apply your process to the search instance to the following set of clauses:

$$\left\{\begin{matrix} \neg P(x,y) \vee P(y,x), \\ P\big(f(x), g(y)\big) \vee \neg R(y), \\ \neg P\big(g(x), f(x)\big), \\ R(x) \vee Q(x,b), \\ \neg Q(a,x) \end{matrix}\right\}$$

Concrete Example: Resolution (VI)

Tasks (cont.):

- Remember its best to think of variables in each clause as independent variables

$$\left\{\begin{matrix} \neg P(x_1, y_1) \lor P(y_1, x_1), \\ P\big(f(x_2), g(y_2)\big) \lor \neg R(y_2), \\ \neg P\big(g(x_3), f(x_3)\big), \\ R(x_4) \lor Q(x_4, b), \\ \neg Q(a, x_5) \end{matrix}\right\}$$

Concrete Example: Resolution (VI)

Tasks (cont.):

- Last two resolved

$$\left\{\begin{array}{c} \neg P(x_1, y_1) \vee P(y_1, x_1), \\ P\big(f(x_2), g(y_2)\big) \vee \neg R(y_2), \\ \neg P\big(g(x_3), f(x_3)\big), \\ \boldsymbol{R(x_4) \vee Q(x_4, b)}, \\ \boldsymbol{\neg Q(a, x_5)}, \\ \boldsymbol{R(a)} \end{array}\right\}$$

mgu = $\{x_4 \approx a, x_5 \approx b\}$

Concrete Example: Resolution (VI)

Tasks (cont.):

- Resolve newest with 2nd

$$\left\{\begin{array}{c} \neg P(x_1, y_1) \vee P(y_1, x_1), \\ \boldsymbol{P\big(f(x_2), g(y_2)\big) \vee \neg R(y_2)}, \\ \neg P\big(g(x_3), f(x_3)\big), \\ R(x_4) \vee Q(x_4, b), \\ \neg Q(a, x_5), \\ \boldsymbol{R(a)}, \\ \boldsymbol{P\big(f(x_2), g(a)\big)} \end{array}\right\}$$

mgu = $\{y_2 \approx a\}$

Concrete Example: Resolution (VI)

Tasks (cont.):

- Resolve newest with first

$$\left\{\begin{array}{c} \neg \boldsymbol{P}(\boldsymbol{x_1}, \boldsymbol{y_1}) \vee \boldsymbol{P}(\boldsymbol{y_1}, \boldsymbol{x_1}), \\ P\big(f(x_2), g(y_2)\big) \vee \neg R(y_2), \\ \neg P\big(g(x_3), f(x_3)\big), \\ R(x_4) \vee Q(x_4, b), \\ \neg Q(a, x_5), \\ R(a), \\ \boldsymbol{P}\big(\boldsymbol{f}(\boldsymbol{x_2}), \boldsymbol{g}(\boldsymbol{a})\big) \\ \boldsymbol{P}(\boldsymbol{g}(\boldsymbol{a}), \boldsymbol{f}(\boldsymbol{x_2})) \end{array}\right\}$$

mgu = $\{x_1 \approx f(x_2), y_1 \approx g(a)\}$

Concrete Example: Resolution (VI)

Tasks (cont.):

- Resolve newest with third

$$\left\{\begin{array}{c} \neg P(x_1, y_1) \lor P(y_1, x_1), \\ P\big(f(x_2), g(y_2)\big) \lor \neg R(y_2), \\ \boldsymbol{\neg P\big(g(x_3), f(x_3)\big),} \\ R(x_4) \lor Q(x_4, b), \\ \neg Q(a, x_5), \\ R(a), \\ P\big(f(x_2), g(a)\big) \\ \boldsymbol{P(g(a), f(x_2))} \\ \blacksquare \end{array}\right\}$$

mgu = $\{x_3 \approx a, x_2 \approx a\}$

Remarks

- Set-based search states can very quickly get very large.
- Usually a lot of extensions are possible
  ☞ control is very important
- Almost all evolutionary search approaches are set-based [see later genetic algorithms]

Onward to …
And-Tree-based Search

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY