Set-based Search

CPSC 433: Artificial Intelligence
Fall 2024

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

August 8, 2024

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.

- Fits most local search problems

Model

Formal Definitions: Model

Set-based Search Model

$A_{set} = (S_{set}, T_{set})$

$\boldsymbol{F}$ set of facts

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

$S_{set} \subseteq 2^{\boldsymbol{F}}$ 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

$T_{set} = \{(s, s') \mid \exists A \rightarrow B \in \boldsymbol{Ext}$ 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

Formal Definitions: Model

Set-based Search Model

$$A_{set} = (S_{set}, T_{set})$$

You need to make

$$\boldsymbol{F}$$

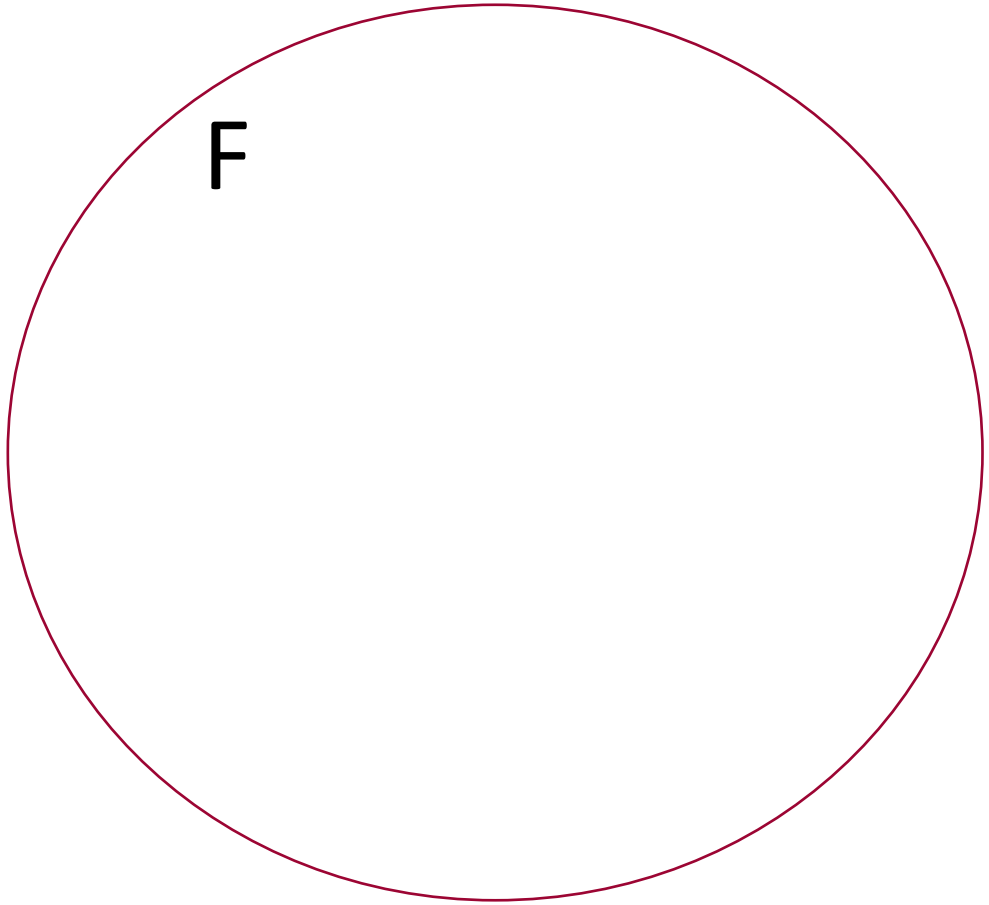
$$\boldsymbol{Ext} \subseteq \{A \rightarrow B \mid A, B \subseteq \boldsymbol{F}\}$$

Comes for free by model definition

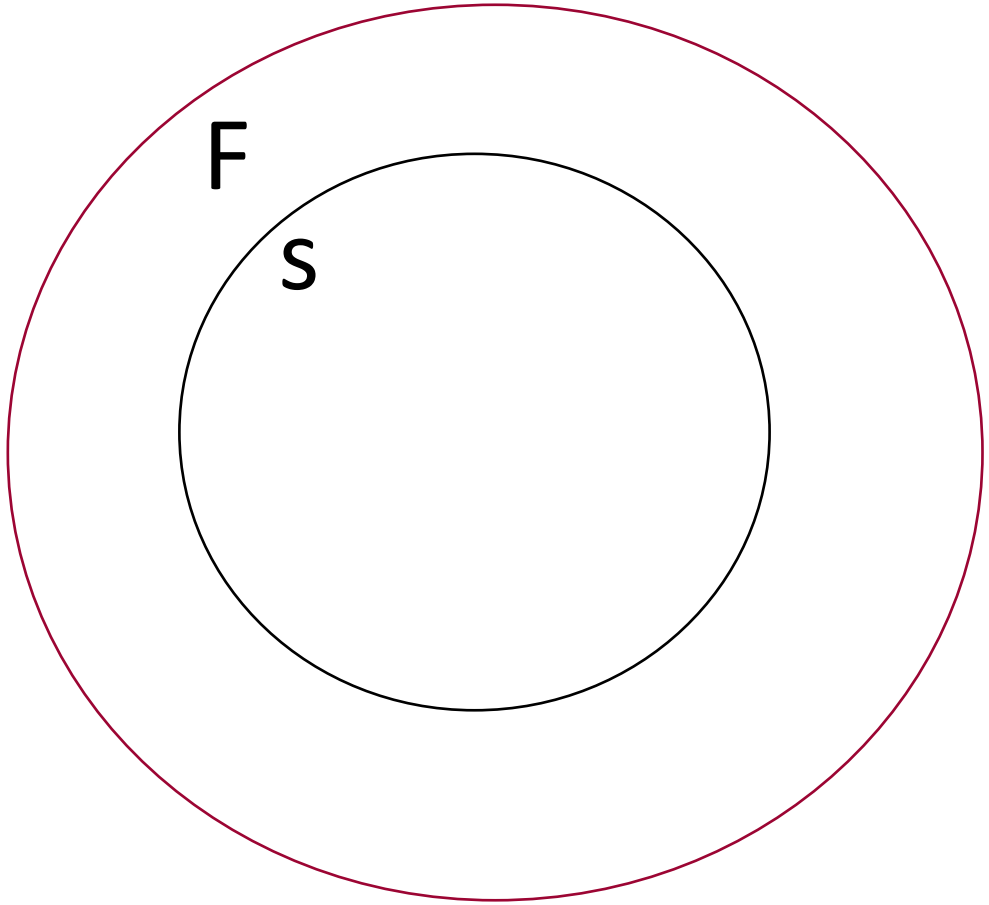
$$S_{set} \subseteq 2^{\boldsymbol{F}}$$

$$T_{set} \subseteq S_{set} \times S_{set}$$

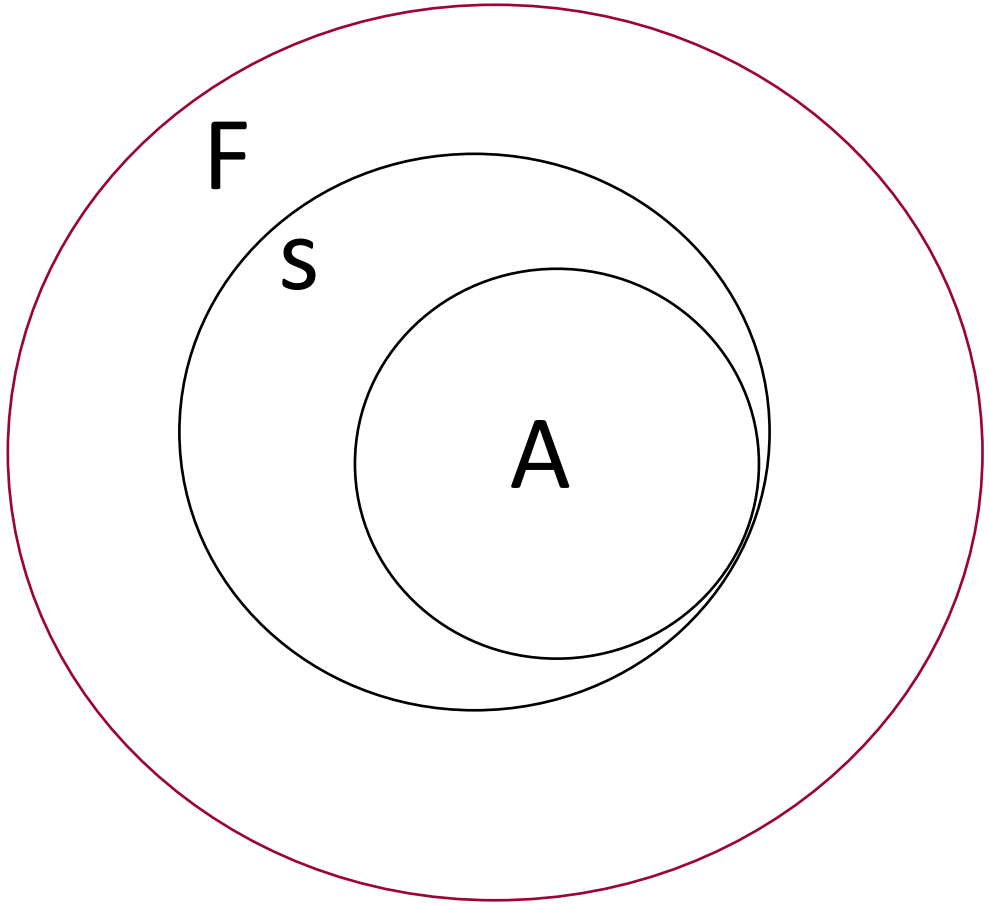
Venn Diagram of Facts and States



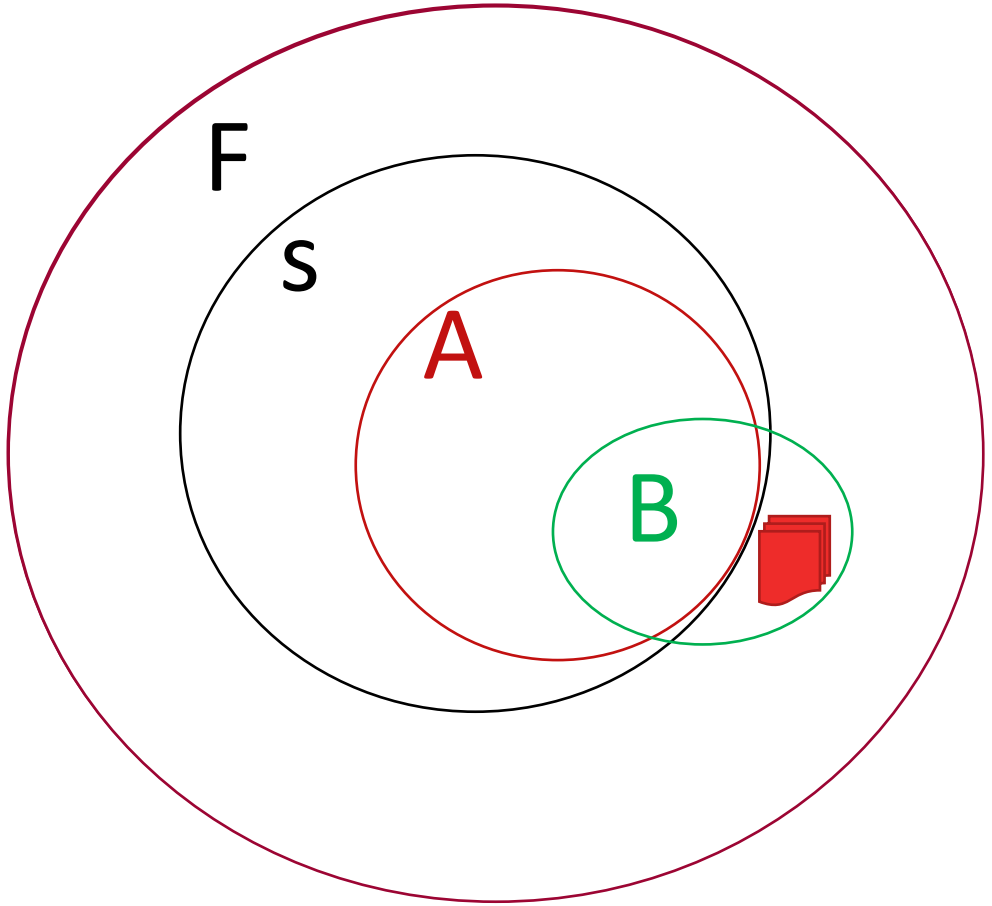
Venn Diagram of Facts and States



Venn Diagram of Facts and States



Venn Diagram of Facts and States

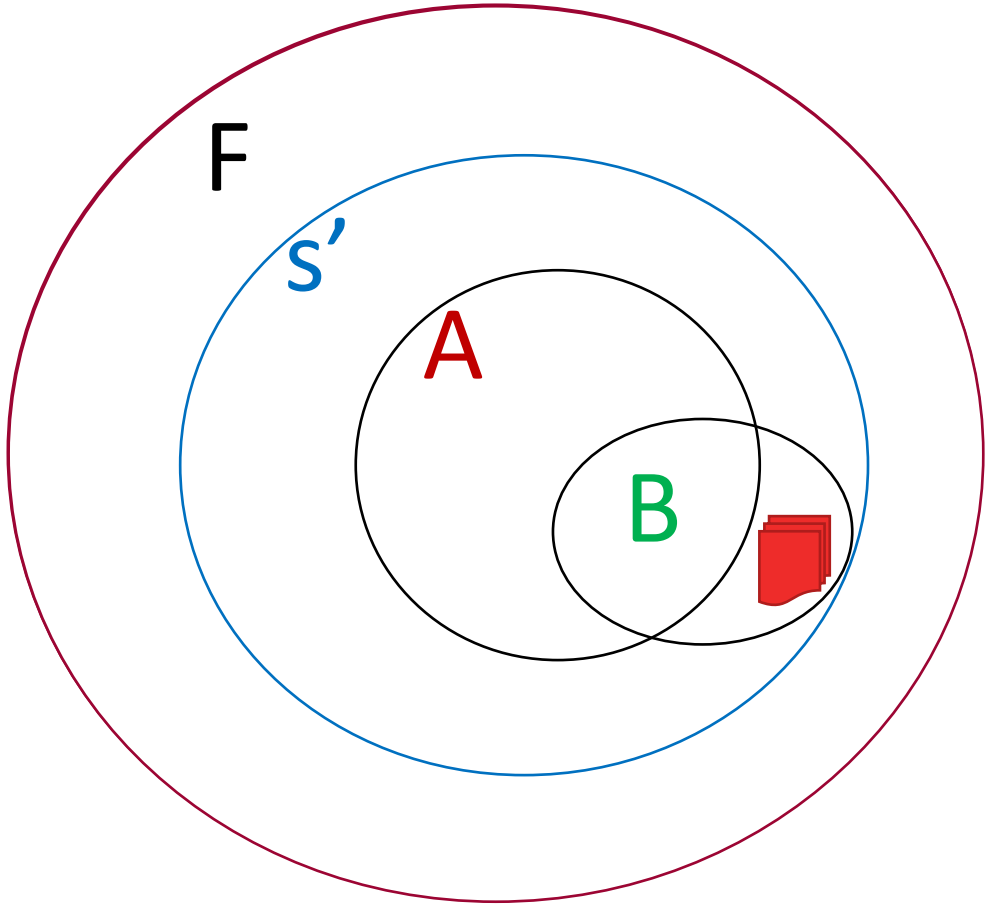


$A \rightarrow B$

B contains new facts!

B may or may not drop facts from A.

Venn Diagram of Facts and States



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

Less formally: Model

- F can consist of solution pieces, solution candidates, parts of a world description, etc.

- With $\mathbf{Ext}$ 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

Process

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 (nothing new yet!)

$K_{set}(s, e) = (s - A) \cup B$ process selects transition rule A->B, where

1. $A \subseteq s$
2. $A \rightarrow B \in \boldsymbol{Ext}$
3. $\forall A' \rightarrow B' \in \boldsymbol{Ext}$ with $A' \subseteq s$ holds: $\boldsymbol{f_{wert}}(A, B, e) \leq \boldsymbol{f_{Wert}}(A', B', e)$
4. $A \rightarrow B = \boldsymbol{f_{select}}(\{A' \rightarrow B' | \boldsymbol{f_{wert}}(A', B', e) \leq \boldsymbol{f_{wert}}(A'', B'', e) \quad \forall A'' \rightarrow B'' \in \boldsymbol{Ext}$ with $A'' \subseteq s\}, e)$

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 (nothing new yet!)

$K_{set}(s, e) = (s - A) \cup B$ process selects transition rule A->B, where

1. A are facts from current state
2. $A \rightarrow B$ is an available extension rule
3. We selected a best rule ($A \rightarrow B$) based on minimizing function f_{wert}
4. Tie breaker f_{select} produces 1 rule($A \rightarrow B$) out of many when multiple f_{wert} are equal

Formal Definitions: Search Process

Set-based Search Process

$$P_{set} = (A_{set}, Env, K_{set})$$

Comes for free by model definition

$K_{set}: S_{set} \times Env \rightarrow S_{set}$ search control is a function K transitioning from current state to next state (nothing new yet!)

$K_{set}(s, e) = (s - A) \cup B$ process selects transition rule A->B, where

You need to make

$f_{wert}: 2^F \times 2^F \times Env \rightarrow \mathbb{N}$ values each choice to a number

$f_{select}: \{2^F \times 2^F\} \times Env \rightarrow 2^F \times 2^F$ 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 $\mathbf{f_{wert}}$)
  - Considering only extensions with minimal evaluation
  - Use $\mathbf{f_{select}}$ as tiebreaker
- Obviously, there usually are many different $\mathbf{f_{wert}}$ and $\boldsymbol{f_{select}}$ functions
- Sometimes $\mathbf{f_{wert}}$ can also produce integers or real numbers

Instance

Formal Definitions: Search Instance

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

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

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

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

Less formally: Search Instance

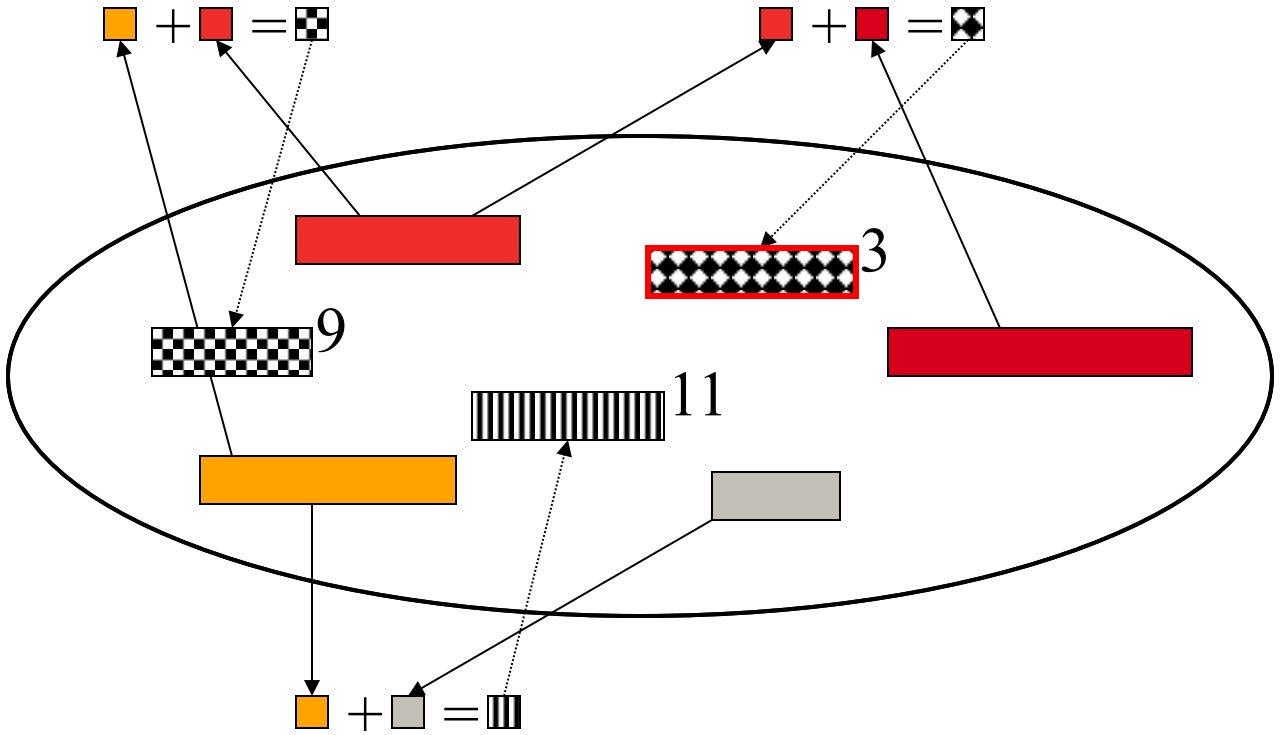
- We start with the given solution pieces, some random solutions, or the given parts of the description (or …)

- 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

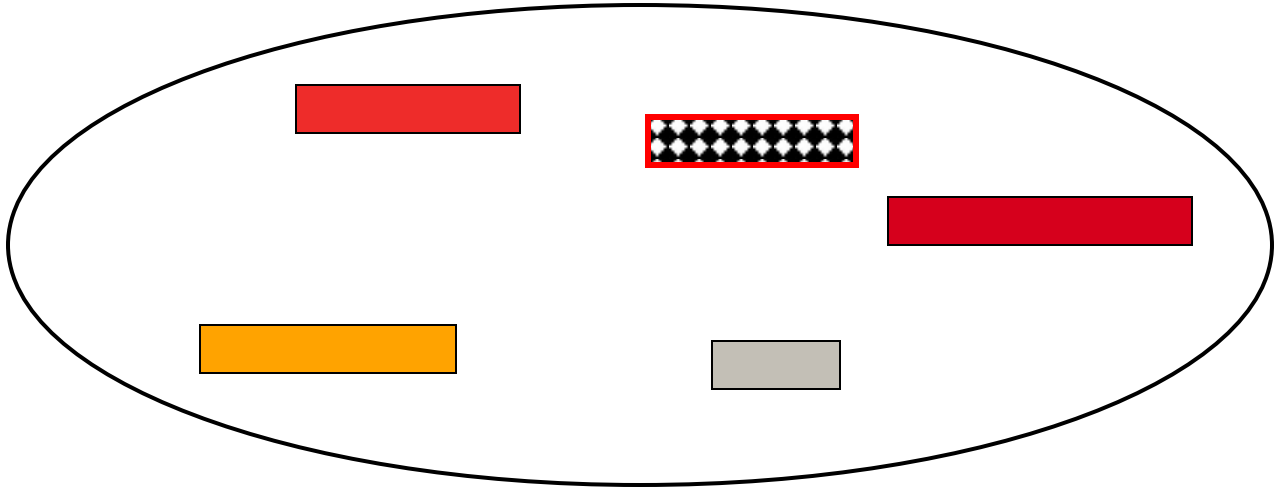
Visualize

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**

- You have your sets **F** and **Ext**. You have now made 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.

- If you want to rely on random decisions (or include them). I.e. not always pick minimal rule. Set f_{wert} constant

5. Identify rules that have the same f_{wert} -value and design f_{select} as tiebreaker (random decisions are best expressed using f_{select})

Review

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 … resolution via set-based search

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY