CHAPTER 16

# Discovering Heuristics with Logic

## 16.1 Discovering Heuristics with Answer Set Programming

Answer set programming, the topic of the previous chapter, has an interesting application beyond single-player games, namely, to discover useful properties of games written in GDL. You can use it to find latches, for example. Recall from Chapter 12 that a latch is a proposition that, when it becomes true (respectively, false), stays so for the rest of the game. To check that a state feature has this property, you first replace the given clauses for the initial state by the following ASP rule.

```
0 { true(F,1) : base(F) }
```

This fact acts as a *state generator* because it supports any base proposition to either be true or not at time 1. The missing number after the closing curly bracket indicates that a model can contain arbitrarily many elements from the set.

We also stipulate that all players choose one legal move at time 1.

```
1 { does(R,M,1) : input(R,M) } 1 :- role(R)

:- does(R,M,1), ~legal(R,M,1)
```

Now suppose that we want to verify an arbitrary base proposition `p` to be a positive latch, that is, to stay true once it becomes true. This is achieved by trying to find a *counter-example*, that is, a model in which `p` is true in one state but not so in the next state.

```
counterexample :- true(p,1), ~true(p,2)
```

Finally, we need to filter out all models without such a counter-example as per the constraint below.

```
:- ~counterexample
```

Every model that an ASP system will generate is a counter-example to the hypothesis that `p` cannot revert back from true to false. In other words, if the solver produces *no* answer, then you have checked that the proposition *is* in fact a latch. This argument involves double negation but is perfectly valid.

In order to check that a proposition cannot revert back to true once it became false, you just need to replace the above by

```
counterexample :- ~true(p,1), true(p,2)
```

Note that you only need two time points for this proof technique, which is one of the reasons why it is very viable in practice.

## 16.2 Goal Heuristics

*Goal heuristics* are based on the idea to evaluate intermediate positions according to their estimated distance to a goal - the closer a state is to our player's goal, the more promising it is. A

good distance measure is often indicative of the quality of domain knowledge that a player has.

In general game playing, players begin with knowing nothing of their goal besides the pure logical description. But this information alone suffices to create a basic distance measure. *Fuzzy Logic* helps us to define it. The main idea is to equate distance with the *degree* to which a given state satisfies a goal formula.

As a motivating example, consider a popular single-player game.

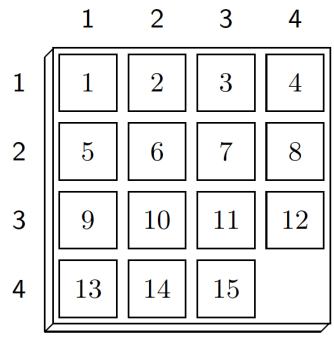



Figure 16.1 - A solved 15-puzzle.

Suppose the GDL description of this game includes the following rule to define the goal.

```
goal(player,100) :-
  true(cell(1,1,1)) & true(cell(2,1,2)) & true(cell(3,1,3)) &
  true(cell(4,1,4)) & true(cell(1,2,5)) & true(cell(2,2,6)) &
  true(cell(3,2,7)) & true(cell(4,2,8)) & true(cell(1,3,9)) &
  true(cell(2,3,10)) & true(cell(3,3,11)) & true(cell(4,3,12)) &
  true(cell(1,4,13)) & true(cell(2,4,14)) & true(cell(3,4,15))
```

Figure 16.2 - The goal formula for the 15-puzzle.

In Fuzzy Logic, the degree to which a conjunctive formula is satisfied is proportional to the number of conjuncts that are true. Hence, an intermediate position in the 15-puzzle will be judged by the number of tiles that are in the correct place. While not a perfect distance measure, it can be successfully employed as the sole parameter of the evaluation function in a depth-limited search with the effect that the player prefers moves that bring it closer to the goal configuration.

## 16.2 Fuzzy Logic

To implement a Fuzzy Logic-based goal heuristics, we first need to fix a truth value $\tau$ with $0.5<\tau<1$. When evaluating a GDL formula against a given state $S$, value $\tau$ will be assigned to all atoms `true(`$p$`)` for which $p$ holds in $S$. Conversely, $1-\tau$ will be assigned to any atom `true(`$p$`)` whose argument $p$ does not hold in $S$.

Consider, for example, a random Tic-Tac-Toe position.

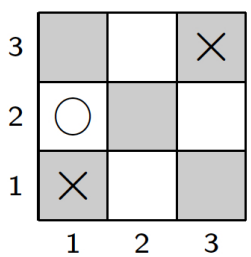



Figure 16.3 - A state in Tic-Tac-Toe.

This state determines the truth values for three selected features shown in the table below, where τ has been set to 0.9.

| Atom | Value |
|---|---|
| `true(cell(1,1,x))` | 0.9 |
| `true(cell(2,2,x))` | 0.1 |
| `true(cell(3,3,x))` | 0.9 |

For the evaluation of compound formulas in Fuzzy Logic, we need to also decide on a so-called *t-norm*. A t-norm (for: *triangular norm*) is a function $T : [0,1] \times [0,1] \rightarrow [0,1]$ that is used to compute the truth value of a conjunction. (As a triangular norm the function must satisfy $T(x,z)>T(y,z)$ whenever $x>y$ and $z>0$.)

A common and simple t-norm is given by the multiplication of the two arguments. For example, by multiplying the individual truth values taken from the table above we can determine the truth value of the conjunction in the formula shown below.

```
diagonal(x) :-
  true(cell(1,1,x)) &
  true(cell(2,2,x)) &
  true(cell(3,3,x))
```

The resulting value is 0.081. This can be interpreted as the degree to which the body of the clause is satisfied in the state of Figure 16.3.

**Evaluating Complex Formulas**

Any t-norm is extensible to an evaluation function for arbitrary formulas with negation (~) and disjunction (|). For our example t-norm, computing the truth value of a compound formula follows the recursive definition shown below.

$$\begin{aligned} \mathit{truth}(\sim A) &= 1 - \mathit{truth}(A) \\ \mathit{truth}(A \;\&\; B) &= \mathit{truth}(A) \cdot \mathit{truth}(B) \\ \mathit{truth}(A \mid B) &= 1 - (1 - \mathit{truth}(A)) \cdot (1 - \mathit{truth}(B)) \end{aligned}$$

The function for the disjunction can be used for the evaluation of an atom that is defined by more than one rule. Consider, for example, the two rules defining a diagonal in Tic-Tac-Toe.

```
diagonal(x) :-
  true(cell(1,1,x)) &
  true(cell(2,2,x)) &
  true(cell(3,3,x))

diagonal(x) :-
  true(cell(1,3,x)) &
  true(cell(2,2,x)) &
  true(cell(3,1,x))
```

The body of the second clause evaluates to $0.1^3 = 0.001$ in the position shown in Figure 16.3. Together with the value for the body of the first clause, 0.081, we obtain $0.081+0.001-(0.081 \cdot 0.001) = 0.081919$ for the truth value of `diagonal(x)`.

Analogously, we can use the rules for `row(M,x)` and `column(N,x)` to compute the truth values for each of their instances, that is, where M,N ∈ {1,2,3}. The resulting values can then be combined into the overall truth value for `line(x)`. Again, this means to compute a disjunction from all instances of all rules for this predicate,

```
line(x) :- row(1,x)
line(x) :- row(2,x)
line(x) :- row(3,x)
line(x) :- column(1,x)
line(x) :- column(2,x)
line(x) :- column(3,x)
line(x) :- diagonal(x)
```

For our example position from Figure 16.3 we thus obtain a truth value of 0.116296 for `line(x)`. In a similar fashion, we can compute the truth value for `line(o)` for the same position as 0.023797. According to the rule,

```
goal(white,100) :- line(x) & ~line(o)
```

we can now compute the degree to which `goal(white,100)` is satisfied.

$$truth(\texttt{goal(white,100)}) = truth(\texttt{line(x)}) \cdot (1 - truth(\texttt{line(o)})) = 0.113529$$

Finally, for games with more goal values than just the maximum of 100, we can compute a weighted average over all goal formulas and use the Fuzzy Logic function for disjunction to obtain a single number for the goal heuristics. In Tic-Tac-Toe, where the two positive goal values are 50 and 100, respectively, we compute the weighted average for white as shown below.

$$\begin{aligned} 100 \cdot [ \quad & truth(\texttt{goal(white,100)}) \cdot 1 \\ + \; & truth(\texttt{goal(white,50)}) \cdot 0.5 \\ - \; & truth(\texttt{goal(white,100)}) \cdot 1 \cdot truth(\texttt{goal(white,50)}) \cdot 0.5 \; ] \end{aligned}$$

With $truth(\texttt{goal(white,50)}) = (1 - truth(\texttt{line(x)})) \cdot (1 - truth(\texttt{line(o)})) = 0.862674$ according to the rule

```
goal(white,50) :- ~line(x) & ~line(o)
```

this altogether results in a goal heuristics value of 49.589684 for the position shown in Figure 16.3.

## 16.3 Using the Goal Heuristics

The primary use of the Fuzzy Logic goal heuristics is to evaluate leaf nodes during a game tree search. To see the heuristics in action, we can apply it to all successor states of the initial position in Tic-Tac-Toe to see which of the possibilities for placing the first mark looks most promising.

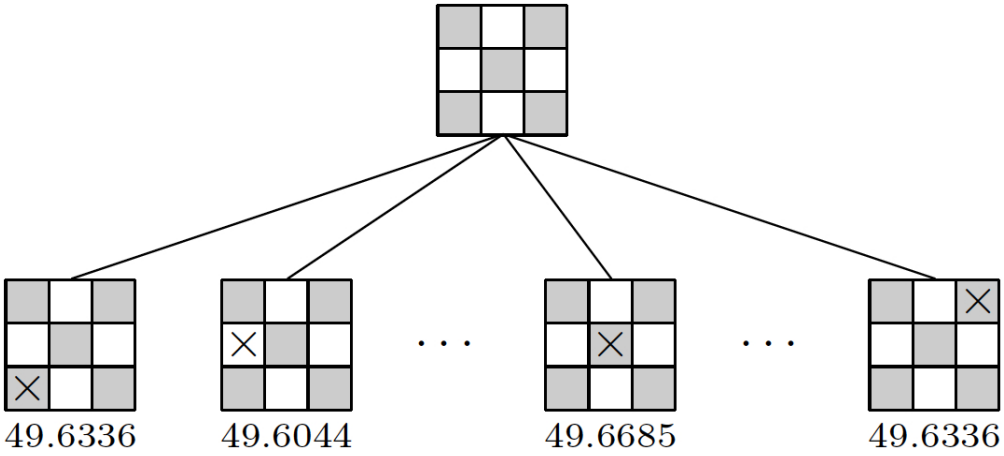



Figure 16.4 - Using the goal heuristics to decide on the opening move in Tic-Tac-Toe.

Without any search beyond the first ply, it follows that the center square is the best move according to the goal heuristics. Moreover, a cell in the corner is deemed more valuable than a non-corner border cell.

Computing the fuzzy truth values for auxiliary atoms, such as `line(x)` or `goal(x,100)`, requires to generate all relevant instances of the rules that define these atoms. This can be easily accomplished with the help of the domain analysis described in chapter 14. The domain graph for Tic-Tac-Toe depicted in Figure 14.2, for example, tells us which instances of the rules for `line(X)` shown below need to be considered when computing the fuzzy truth value of `line(x)` in any given situation.

```
line(X) :- row(M,X)
line(X) :- column(N,X)
line(X) :- diagonal(X)
```

## 16.4 Optimizations and Limitations

When computing the fuzzy truth value of a defined predicate, atoms in the body of a rule can be treated differently if their truth is independent of the current state. Examples include instances of the keywords `role` and `distinct` as well as every auxiliary predicate whose definition does not depend on keyword `true`. Any such state-independent atom can be assigned truth value 1 (if it is true) or 0 (if it is false) rather than $\tau$ or 1-$\tau$. So doing simplifies the computation and also leads to sharper distinctions between different positions.

The strict application of our example t-norm for the goal heuristics can have the practical disadvantage of approaching 0 for large conjunctions even when each conjunct is true. The goal predicate in the 15-puzzle shown in Figure 16.2, for instance, still has a low degree of $\tau^{15} = 0.9^{15} = 0.20589$ after the final goal position has been reached. In practice it is therefore useful to introduce a *threshold* $\theta$ that satisfies $0.5 < \theta \leq \tau$. This threshold can be used to ensure that a true formula always has a truth value greater than 0.5. To do so, we just need to slightly extend the computation of the fuzzy truth values for complex formulas.

$$
\begin{aligned}
truth(\sim A) &= 1 - truth(A) \\
truth(A \;\&\; B) &= max\{truth(A) \cdot truth(B); \theta\} && \text{if } truth(A) > 0.5, truth(B) > 0.5 \\
truth(A \;\&\; B) &= truth(A) \cdot truth(B) && \text{otherwise} \\
truth(A \mid B) &= 1 - (1 - truth(A)) \cdot (1 - truth(B))
\end{aligned}
$$

The better a heuristics can distinguish between different states, the more useful it is for evaluation functions. The multiplication t-norm allows for little flexibility in this regard. A much wider range

of t-norms is captured by the so-called *Yager family*. Each t-norm from this family is given for $0 \leq q \leq \infty$ by

$$S(x,y) = (x^q + y^q)^{1/q}$$
$$T(x,y) = 1 - S(1-x,1-y)$$

Any function S that is used to define a Yager t-norm T also serves as the corresponding computation rule for disjunctions. General game-playing systems that use these t-norms can adjust parameter $q$ as well as the basic truth value $\tau$ depending on the complexity and structure of the goal formulas for different games.

### Limitations

The Fuzzy Logic-based goal heuristics can be very useful for games whose goal, broadly speaking, is reachable through achieving successive sub-goals. This extends well beyond obvious examples like the 15-puzzle shown above. If, for instance, the aim is to eliminate all of your opponent's pieces or to occupy all locations on a board, and if this can be achieved successively, then goal heuristics can provide a useful guidance for a depth-limited search.

A limitation of the simple goal heuristics is not to take into account how difficult it is to extend a partial solution to a complete one, or even if that is possible at all. In Tic-Tac-Toe, for example, every row, column, or diagonal with exactly two of our player's markers has the same fuzzy truth value, no matter whether the remaining third square is still free or already blocked. A possible solution could involve the concepts of latches and inhibitors (cf. Chapter 12). A proposition that is inhibited by an active latch can always be assigned truth value 0 because, by definition, it will never become true later in the game.

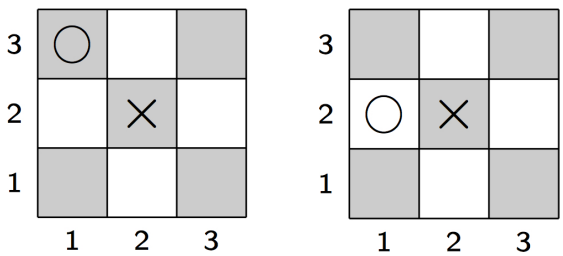
Simple goal heuristics moreover generally fail to provide useful information for games with specialized goals like, for example, checkmate, whose logical definition alone does not allow to determine the degree to which it is already satisfied in an intermediate position.

## Exercises

1. Use answer set programming to discover the following properties in the Buttons and Lights game of Chapter 15.

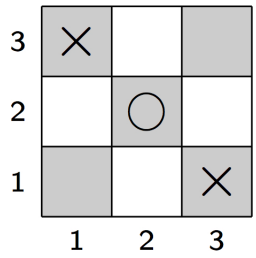
    a. Verify that `step(1)` is a (negative) latch.
    b. Show that none of the propositions `p, q, r`, which represent the status of the lights, is a latch in this game.
    c. Write a module for your player that uses an ASP system to systematically check every base proposition whether it is a latch.

2. Consider the two possible replies shown below to white's opening move.

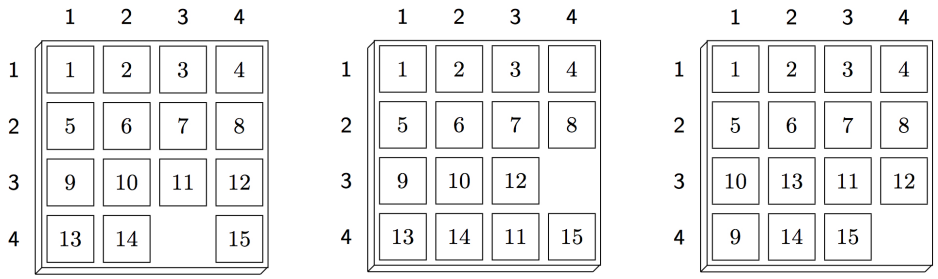



    What truth value does `goal(black,100)` have in these two positions? Which move, therefore, would black take based on the goal heuristics without search?

3. If both players always choose their moves based on the goal heuristics without search, how would the game be played out starting in the following position, where black moves next?




4. For each of the three positions shown below, what is the truth value of the formula in the body of the goal description for the 15-puzzle (cf. Figure 16.2)? Use the multiplication as t-norm to answer this question, with truth value $\tau = 0.9$ and no threshold $\theta$.




5. Answer the previous question for the same three positions, the same t-norm and truth value $\tau = 0.9$, but now using a threshold $\theta$ of 0.6.

6. Implement the Fuzzy Logic goal heuristics in your player and try it out on the 15-puzzle using minimax search.