Strategies

Notes related to design problems and strategies.

Z3

Z3 looks for models that satisfy a set of specified constraints.

- Input parameters determine constraints on frames, models, and worlds.
- Each sentence of the object language will also determine a further constraint on models and worlds.
- Any Z3 model that satisfies the constraints for a set of sentences together with the general constraints given by the input parameters may then be stored in an output file.

Input Parameters

Variables to be specified by the user include:

1. Let `N` be the number of atomic states under consideration.
2. Let `Atoms` be a finite set of sentence letters, e.g., $\{A, B, C, D\}$.
3. Let `Gamma` be a finite set of sentences with sentence letters in `Atoms`.

Glossary

These definitions indicate the intended meanings of the elements employed blow.

1. `State(x)` includes the predicate `State` whose extension is to be determined by Z3 given the constraints below.
2. `Or(x,y)` indicates the bitwise operation which takes the greatest value for each index of the bit vectors `x` and `y`.
3. `Possible(x)` includes the predicate `Possible` whose extension is to be determined by Z3 given the constraints below.
4. `Proposition(X)` is a set of Z3 constraints that require `X` to be a proposition.
5. `Semantics(w,X)` is a set of Z3 constraints that require `X` to be true in `w`.
6. `Alternative(u,w,X)` is a set of constraints that require `u` to be a (perhaps non-unique) result of minimally changing `w` to include a verifier for `X`.

Frame Constraints

The following constraints only depend on `N`, and are always to be included.

1. If `State(x)`, then `BitVec(x,N)` i.e., 'x is a bitvector of length N'.
2. If `State(x)` and `State(y)`, then `State(Or(x,y))`.
3. If `Possible(x)`, then `State(x)`.
4. If `Possible(x)`, `State(y)`, and `Or(x,y) = x`, then `Possible(y)`.

Model Constraints

The following constraints depend on the functions `Proposition(X)` and `Semantics(w,X)` defined in the following section for Python.

1. For each sentence letter `X` in `Atoms`, include the output of `Proposition(X)`.
2. `Possible(w)` and `Or(x,w) = w` whenever `Possible(x)` and `Possible(Or(x,w))`.
3. For each sentence `X` in `Gamma`, include the output of `Semantics(w,X)`.

Python

Python will provide translations between readable sentences and Z3 model constraints as well as between Z3 models and readable model structures.

Infix/Prefix Translators

1. Given sentences in infix notation (e.g., `(A \wedge B) \boxright C`), the prefix translator will output a unique sentences in prefix notation using lists (e.g., `[\boxright, [\wedge, A, B]]`).
2. Given a sentence in prefix notation, the infix translator will output a unique sentence in infix notation.
3. For any sentence `A` in infix notation, `A = infix(prefix(A))`.

Proposition Constraints

Given any sentence letter `X` in `Atoms`, the function `Proposition(X)` yields the following Z3 constraints as outputs:

1. If `Verify(x,X)` or `Falsify(x,X)`, then `State(x)`.
2. If `Verify(x,X)` and `Verify(y,X)`, then `Verify(Or(x,y,X))`.
3. If `Falsify(x,X)` and `Falsify(y,X)`, then `Falsify(Or(x,y,X))`.
4. If `Verify(x,X)` and `Falsify(y,X)`, then not `Possible(Or(x,y))`.
5. If `Possible(x)`, then `Possible(Or(x,y))` where either `Verify(y,X)` or `Falsify(y,X)`.

The constraints on `Verify` and `Falsify` may be extended to complex sentences by way of the following general constraints which are always to be included:

1. If `Verify(z,[\neg, X])`, then `Falsify(z,X)`.
2. If `Falsify(z,[\neg, X])`, then `Verify(z,X)`.
3. If `Verify(z,[\wedge, X,Y])`, then `z = Or(x,y)` where `Verify(x,X)` and `Verify(y,Y)`.
4. If `Falsify(z,[\wedge, X,Y])`, then `Falsify(z,X)` or `Falsify(z,Y)` (or both).
5. If `Verify(z,[\vee, X,Y])`, then `Verify(z,X)` or `Verify(z,Y)` (or both).
6. If `Falsify(z,[\vee, X,Y])`, then `z = Or(x,y)` where `Falsify(x,X)` and `Falsify(y,Y)`.

World Alternatives

Given a world `w` and sentence `X`, the function `Alternatives(u,w,X)` yields the following Z3 constraints:

1. `State(x)` where `Verify(x,X)` and `Or(x,u) = u`.
2. `State(y)` where `Or(y,w) = w` and `Or(y,u) = u`.
3. If `State(z)`, `Or(z,w) = w`, `Possible(Or(x,z))`, and `Or(y,z) = z`, then `y = z`.

Semantic Constraints

`Semantics(w,X)` may be defined recursively as follows:

1. If `X` is in `Atoms`, then `Semantics(w,X)` is: `Verify(x,X)` and `Or(x,w) = w`.
2. If `X = [\neg,Y]`, then `Semantics(w,X)` is: not `Semantics(w,Y)`.
3. If `X = [\wedge,Y,Z]`, then `Semantics(w,X)` is: `Semantics(w,Y)` and `Semantics(w,Z)`.
4. If `X = [\vee,Y,Z]`, then `Semantics(w,X)` is: `Semantics(w,Y)` or `Semantics(w,Z)`.
5. If `X = [\Box,Y]` and neither `\Box` nor `\boxright` occur in `Y`, then `Semantics(w,X)` is: `Semantics(u,Y)` whenever `World(u)`.
6. If `X = [\boxright,Y,Z]` and neither `\Box` nor `\boxright` occur in `Y`, then `Semantics(w,X)` is: `Semantics(u,Z)` whenever `Alternative(u,w,Y)` and `World(u)`.

Model Builder

Z3 models will be translated back into a natural representation.

- The python representation of a Z3 model should specify the states, possible states, worlds, and the propositions assigned to the sentence letters in question.
- A function should specify the propositions assigned to all subsentences of the sentences under evaluation.
- These details may then be stored in an output file, prompting the user whether to search for another model.