Missing Assumption Search

by Sven Nilsen, 2018

In this paper I formalize a class of useful problems that is strictly harder than automatic theorem proving. This class can be thought of as starting with a list of assumptions and an unprovable proof goal and then find one or more missing assumptions such that the goal becomes provable. This is less complex than reverse mathematics and a superset of abductive reasoning.

Formal mathematics has a curious property that any mathematical object can be constructed from some knowledge about the formal grammar of the mathematical language. In many problems, we only have a partial description of the full picture available and it is required to fill out some missing piece on our own. This missing piece of information is filled out by having knowledge of the general language which the problem is described. With other words, it is equivalent to finding a missing assumption.

A missing assumption search problem has the following properties (using colors for locations):

A => B	start => goal
C	missing assumption
$A \wedge C \to B$	\`B\` is provable from \`A\` and \`C\`
$A \to B \vee C$	If \`A\` is true then either \`B\` or \`C\` is true

The second constraint might seem unnecessary at first, because in a standard example of logic:

$$A \wedge (A \to B) \to B$$
$$A \wedge C \wedge (C = (A \to B)) \to B$$
$$A \wedge C \to ((C = (A \to B)) \to B)$$

However, the context is not \`A ∧ C\` but \`A ∧ C → B\`. The following holds only when \`A → B ∨ C\`:

$$(A \wedge C \to B) \to ((C = (A \to B)) \to B)$$
$$(A \wedge (A \to B) \to B) \to B$$

The irony here is that one does a missing assumption search to find the missing assumption for missing assumption search. Luckily, this can be done by brute force and transforming the failure case:

$$\neg(A \wedge \neg B \wedge \neg C)$$
$$\neg A \vee B \vee C$$
$$\neg A \vee (B \vee C)$$
$$A \to B \vee C$$

There are three trivial problems where there are no need to find the missing assumption:

A = false	From falsehood one can prove anything
B = true	\`B\` is always provable (tautology)
$(A = \text{true}) \wedge (A \to B)$	There is no missing assumption if one can prove \`B\` from \`A\`

Any of the three trivial problems implies that the missing assumption search is solved:

$(A = \text{false}) \rightarrow (A \wedge C \rightarrow B) \wedge (A \rightarrow B \vee C)$
$(B = \text{true}) \rightarrow (A \wedge C \rightarrow B) \wedge (A \rightarrow B \vee C)$
$(A = \text{true}) \wedge (A \rightarrow B) \rightarrow (A \wedge C \rightarrow B) \wedge (A \rightarrow B \vee C)$

The non-trivial problems are when `A` is `true`, `B` is not a tautology and `B` is not provable from `A`.

When writing a problem in the form (using colors to emphasize locations instead of variables):

A => B	start => goal
C	missing assumption

It is usually assumed that there is a non-trivial problem and `C` must be found such that:

$(A \wedge C \rightarrow B) \wedge (A \rightarrow B \vee C) \rightarrow ((C = X) \rightarrow (A \rightarrow B))$

When `C` is assigned `X`, `B` becomes provable from `A`.

For example:

A => B	start => goal
A → B	missing assumption

In the example above, it is not provable that `B` follows from `A`. By adding the assumption `A → B`, it becomes provable.

Since assuming the goal always implies the goal, it is undesirable to consider the goal as a missing assumption, because it might make other existing assumptions irrelevant.

A => B	start => goal
B	missing assumption (bad guess, because `A` becomes irrelevant)

Another example:

A → B => B	start => goal
A	missing assumption

In the example above, adding the assumption `A` makes the goal provable.

Here is another example:

(A → B) ∧ C => B	start => goal
C → A	missing assumption

Instead of choosing `A` or `C → B` as the missing assumption above, `C → A` is chosen to utilize both terms in the start assumption. `A` implies `C → A` but not vice versa, so `C → A` is a weaker assumption. `C → B` would make `A → B` irrelevant. When there is a preference order among possible missing assumptions like this, the search corresponds to abductive reasoning.

It is worth noticing that missing assumption search is not limited to logic, but can be applied to any formal language. In logic, provability is represented as material implication `→`, but this is not true for all formal languages. There can be a formal language that has no `→` operator, but where provability is represented implicitly by an automatic theorem prover.

Abductive reasoning is similar to missing assumption search, except that there is a preference order to the candidates that makes the start assumptions provable. In general, there is a tendency of natural occurring preference order:

- Avoid “short-circuiting” of terms in start assumption by making them irrelevant
- Adding weaker assumptions is better than strong assumptions
- Adding simple assumptions is better than complex assumptions
- Avoid adding new variables
- Reuse variables from start assumption

The argument against short-circuiting is motivated by a recursive meta-perspective of missing assumption search. When short-circuiting, the search problem can be simplified to an equivalent but less complex search problem. If this was desirable, then why not just simplify the problem upfront? A meta-assumption is made that only start assumptions that are relevant for the search problem are used. This should make the search problem irreducible after adding the missing assumption, at least under ideal circumstances. One can imagine two agents communicating in a way to make abductive reasoning easier, which implies that the agents only speak relevant information to the context/situation.

Another argument against short-circuiting is that the more flexible formal language that is used for automatic theorem proving, the more opportunities are there for short-circuiting. This introduces a problem where extending a formal language with new features leads to different behavior than previous versions. Avoiding short-circuiting makes the behavior predictable for future versions.

The argument for weaker assumptions is that one usually wants the assumptions to describe as many mathematical objects as possible. A strong assumption limits the set of possible mathematical objects more than a weak assumption.

An argument for simple assumptions is that one does not want to clean up descriptions of equivalent terms afterwards. The simpler expressions, the more likely it is to match the same expression that the user would use if the missing assumption was known. People tend to use short expressions for common concepts and build these sugared representations into formal language both to make it more readable and improve performance of inference. One method to use simple assumptions is to restrict generation to expressions a maximum length.

In some formal languages, such as first-order logic, it is possible to add expressions that introduces new variables. This is usually not desired unless there is a trade-off with simplicity or shorter description. The argument against new variables is the same as for simple assumptions.

An argument for reusing variables from start assumption is to find desirable candidates more quickly. Many inference rules of formal languages are abstracted over variables, so by reusing variables it is more likely that an attempt will make the goal provable. This requires traversing the expressions of the start assumption and extract variables before generating candidates for missing assumptions.

The reason missing assumption search is strictly harder than automatic theorem proving is the following: If checking for provability of the goal requires an automatic theorem prover, then the problem of finding the missing assumption is a higher order problem taking the automatic theorem prover as an argument. During the search, the higher order solver needs to call the concrete solver multiple times. It means the search problem is strictly harder than attempting to prove the goal from some assumption.

In reverse mathematics, one starts with a theory and then attempts to find axioms for which the theory is provable. This does not require any start assumption, but considers all input as part of the proof goal. Since missing assumption search requires only finding a missing piece instead of all assumptions, it is less complex than reverse mathematics.

Missing assumption search is a superset of abductive reasoning because abductive reasoning makes an additional assumption of “best explanation”. While the usage of these two overlaps, an algorithm for missing assumption search might be simpler because it can run until it finds a match satisfying some prepared constraints and then stop. The user might not know what it means to find the best explanation and try different settings in an iterative way, learning from the output of the solver.

Finally, here is an example how missing assumption search works in a custom formal language:

> has(Alice, Ball), gives(Alice, Bob, Ball), gives(Carl, Dexter, Ball) => has(Dexter, Ball)
> gives(Bob, Carl, Ball)

Alice has the ball and gives it to Bob. Now, Bob has the ball. However, there is no action where Bob gives the ball to Carl, so Carl can not give it further to Dexter. By adding the missing action where Bob gives the ball to Carl, Carl gets the ball and can give it to Dexter. Finally, Dexter has the ball.

This formal language is not made of traditional logic, but requires changing states of `has` during the proof. The point is to demonstrate that missing assumption search is more general than logic.