Model Based Projection for Quantifier Instantiation

No Author Given

No Institute Given

1 Problem Statement

Let $\varphi = G \wedge \forall \vec{x}.\ F(\vec{x})$ be a closed formula, where G, F are quantifier-free.

The goal is to determine (un)satisfiability of φ by iteratively instantiating $\forall \vec{x}.\ F(\vec{x})$. We denote instantiations of $\forall \vec{x}.\ F(\vec{x})$ by F_i.

Definition 1 (Syntactic instantiation). *Given ground terms $\vec{t}$, the formula $F[\vec{t}/\vec{x}]$ is a* syntactic instantiation *of $\forall \vec{x}.\ F(\vec{x})$.*

Clearly, $\forall \vec{x}.\ F(\vec{x}) \implies F[\vec{t}/\vec{x}]$.

Remark 1. Instantiations are implied by $\forall \vec{x}.\ F(\vec{x})$, but the computation of $\vec{t}$ may depend on G as well (this is the case in the classical MBQI).

Definition 2. *A ground formula F_i such that $\forall \vec{x}.\ F(\vec{x}) \implies F_i$ is a* semantic instantiation *of $\forall \vec{x}.\ F(\vec{x})$.*

The important property of instantiations is:

Observation 1 (Soundness) *If $\{F_i\}_i$ are semantic instantiations of $\forall \vec{x}.\ F(\vec{x})$, and $G \wedge \bigwedge_i F_i$ is unsatisfiable, then so is $G \wedge \forall \vec{x}.\ F(\vec{x})$.*

1.1 Relation to MBP

Definition 3 (Model based projection). *Given a quantifier-free formula $H(\vec{x})$, a model M and an assignment $\vec{v}$ to $\vec{x}$ such that $M, \vec{v} \models H(\vec{x})$, a* model-based projection *of $H(\vec{x})$ w.r.t. $(M, \vec{v})$ is a ground formula, denoted $Mbp(H(\vec{x}), M, \vec{v})$, such that*

- *$M \models Mbp(H(\vec{x}), M, \vec{v})$, and*
- *$Mbp(H(\vec{x}), M, \vec{v}) \implies \exists \vec{x}.\ H(\vec{x})$.* [*Sharon:* In the mbi work, the projections we define do not necessarily satisfy this. For example, if $H(x) := p(x)$, then the projection is $\top$, even though $\top \not\Longrightarrow \exists x.\ p(x)$. All we are guaranteed is a weaker property: if M satisfies the projection, then it can be adapted to a model of H while preserving all equalities over *shared terms* (in this example, x is not equivalent to any shared term)]

Observation 2 *If $M \not\models \forall \vec{x}.\ F(\vec{x})$, and $\vec{v}$ is a witness assignment, i.e., $M, \vec{v} \not\models F(\vec{x})$ (equivalently, $M, \vec{v} \models \neg F(\vec{x})$), then $\neg Mbp(\neg F(\vec{x}), M, \vec{v})$ is a semantic instantiation of $\forall \vec{x}.\ F(\vec{x})$.*

This is because $Mbp(\neg F(\vec{x}), M, \vec{v}) \implies \exists \vec{x}.\ \neg F(\vec{x})$ iff $\forall \vec{x}.\ F(\vec{x}) \implies \neg Mbp(\neg F(\vec{x}), M, \vec{v})$ (contraposition).

1.2 Covers and Consequences

The appropriate requirement for interpolation seems to be to incrementally compute a *cover* $\bigvee_i \Psi_i$, such that

$$\exists \vec{x}.\ F[\vec{x}] \implies \bigvee_i \Psi_i.$$

The cover should satisfy

$$G \wedge \exists \vec{x}.\ F[\vec{x}] \quad \text{is sat} \quad \text{iff} \quad G \wedge (\bigvee_i \Psi_i) \quad \text{is sat.}$$

For MBQI, a dual requirement seems more adequate. The requirement is to compute consequences Φ_i, such that

$$\forall \vec{x}.\ F[\vec{x}] \implies \bigwedge_i \Phi_i,$$

and for context G:

$$G \wedge \forall \vec{x}.\ F[\vec{x}] \quad \text{is sat} \quad \text{iff} \quad G \wedge (\bigwedge_i \Phi_i) \quad \text{is sat.}$$

Model-based skolemization for arithmetic can be used for both purposes because it eventually produces a disjunction equivalent to $\exists \vec{x}.\ F[\vec{x}]$. But it does not mean that any procedure that is adequate for MBI is adequate for MBQI. Take for example the formula $F[x] = p(x)$, where p is an uninterpreted predicate. An adequate MBI cover for $\exists x.p(x)$ is $\top$ regardless of the quantifier free context where this formula resides. For example $(\exists x.p(x)) \wedge \neg p(a) \wedge \neg p(b)$ is satisfiable whenever the domain of x can have more than two elements. The dual formula $(\forall x.p(x)) \wedge \neg p(a) \wedge \neg p(b)$ is clearly not satisfiable.

2 Array Property Fragment (APF)

An array property is a formula of the form $\forall \vec{i}.\ \text{grd} \rightarrow \text{val}$ with:

$$\begin{aligned} \text{grd} &::= \text{grd} \wedge \text{grd} \mid \text{grd} \vee \text{grd} \mid \text{atom} \\ \text{atom} &::= \text{expr} \leq \text{expr} \mid \text{expr} = \text{expr} \\ \text{expr} &::= i \mid t \end{aligned}$$

where $i \in \vec{i}$ and t is a linear term that does not include any variable from $\vec{i}$.

The formula val is restricted to include variables from $\vec{i}$ only in terms of the form $a[i]$, and to not have any nested array terms $(a[a[i]])$.

An APF formula is any Boolean combination of array properties as above and quantifier-free array formulas.

Decision procedure for APF. Roughly:

- eliminate writes
- skolemize existentially quantified variables
- replace each universal quantifier $\forall \vec{i}.\ \psi(\vec{i})$ with a finite conjunction of instantiations of ψ over the set of index terms: $\bigwedge_{\vec{b} \in \mathcal{I}} \psi[\vec{b}/\vec{i}]$.

The set $\mathcal{I}$ of index terms used for instantiation is defined based on the formula obtained after write elimination and skolemization as follows:

- ground boundary terms used in guards (these are the t terms from the grammar),
- ground terms at which an array is read, and
- 0 if there are no terms as above.

After the transformation, arrays are treated as uninterpreted functions.

Claim. The formula resulting from the transformation is equi-satisfiable to the original formula.

Proof (sketch). It suffices to show that the instantiation step preserved (un)satisfiability. Let φ denote the formula before the instantiation step, and φ' the formula after instantiation. Clearly, $\forall \vec{i}.\ \psi(\vec{i})$ implies $\bigwedge_{\vec{b} \in \mathcal{I}} \psi[\vec{b}/\vec{i}]$, hence satisfiability of φ implies satisfiability of φ'.

As for the opposite direction, let M be a model of φ'. Define a projection function *proj* over integer values that maps $v \in \mathcal{Z}$ to its least upper bound from $\{M(b) \mid b \in \mathcal{I}\}$ or to its greatest lower bound if no least upper bound exists. Now define M' as M except that for the interpretation of array variables, which is adapted: $M'(a)(\vec{v}) = M(a)(\vec{proj}(v))$. That is, M' partitions a to ranges, where each range of indices is associated by *proj* with some $b \in \mathcal{I}$ and the value stored in all the indices in the range is $M(a[b])$, i.e., each index in the range associated with b "behaves" like b. Since the interpretation of the terms in φ that read from arrays remains the same under M and M' (since all terms where an array is read are in $\mathcal{I}$), and all qualities/disequalities between arrays are preserved, $M' \models \varphi'$ as well.

Finally, we show that $M' \models \varphi$. Suppose $M' \models \bigwedge_{\vec{b} \in \mathcal{I}} \psi[\vec{b}/\vec{i}]$, and show that $M' \models \forall \vec{i}.\ \psi(\vec{i})$, where $\psi(\vec{i}) = \psi_g(\vec{i}) \to \psi_v(\vec{i})$. If $M', \vec{v} \models \psi_g(\vec{i})$, then $M', \vec{proj}(v) \models \psi_g(\vec{i})$ as well (due to the definition of projection that preserves inequalities). Let $\vec{b} \in \mathcal{I}$ be such that $\vec{proj}(v) = M(\vec{b}) = M'(\vec{b})$. Then $M' \models \psi_g[\vec{b}/\vec{i}]$. Hence, $M' \models \psi_v[\vec{b}/\vec{i}]$, which again means $M', \vec{proj}(v) \models \psi_v(\vec{i})$. Relying on the fact that in M' all indices behave like projected indices, we have that $M', \vec{v} \models \psi_v(\vec{i})$.

3 MBQI via projection

Fix a formula $G \wedge \forall \vec{x}.\ \neg F(\vec{x})$ with a single unary uninterpreted function symbol $g(\cdot)$ whose argument is of sort integer. (Assume that there exists at least one ground term $g(a)$.)

Let $M_0 \models G$. First, we extend M_0 into a model M that also interprets $\vec{x}$ such that $M \models F(\vec{x})$, if exists (otherwise, $M_0 \models G \wedge \forall \vec{x}.\ \neg F(\vec{x})$ and we are done). To extend M_0, we start by constructing a representation of M_0.

Specializing a formula to M_0. There exists a set $s_0, \ldots, s_k$ of ground terms that appear in terms of the form $g(s_i)$ in G, do not include g, are ordered such that $M_0(s_i) < M_0(s_{i+1})$ for every $0 \leq i < k$ and $M_0(g)$ maps every $v \in (M_0(s_i), M_0(s_{i+1})]$ for $0 \leq i < k$ to $M_0(g(s_{i+1}))$, every $v \leq M_0(s_0)$ to $M_0(g(s_0))$, and every $v > M_0(s_k)$ to $M_0(g(s_k))$.

- Denote by g^{M_0} the symbolic expression that encodes the interpretation of g in M_0. It is obtained by symbolically representing the intervals above and their images using ite constructs; the boundaries of the interval $(M_0(s_i), M_0(s_{i+1})]$ are represented by s_i and s_{i+1}, and the image of the function, $M_0(g(s_{i+1}))$, is represented by $g(s_{i+1})$. (In case $n = 1$, $M_0(g)$ maps any integer to $M_0(g(s_1))$, and g^{M_0} is represented accordingly.)
- Denote by F^{M_0} the specialization of $F(\vec{x})$ to M_0, where (i) every term $g(t)$ is substituted by $g^{M_0}(t)$ (recursively, top-down); hence, F^{M_0} no longer includes g, and (ii) every uninterpreted arithmetic constants is substituted by its interpretation in M_0.

The only uninterpreted symbols in F^{M_0} are $\vec{x}$.

Claim. $F^{M_0}(\vec{x})$ is satisfiable iff $M_0 \models \exists \vec{x}.\ F(\vec{x})$. Furthermore, $M_1 \models F^{M_0}(\vec{x})$ iff $M_0 \cup M_1 \models F(\vec{x})$.

When $F^{M_0}(\vec{x})$ is satisfiable, M_1 as above allows us to extend M_0 to a model $M = M_0 \cup M_1$ of $F(\vec{x})$. We can then use Mbs on $M \models F(\vec{x})$ to extract ground terms $\vec{w}$ that will be used in a new instance $\neg F(\vec{w})$ of $\forall \vec{x}.\ \neg F(\vec{x})$. However, we cannot apply Mbs on F directly. We first need to transform it into a formula F' where there is no occurrence of $\vec{x}$ in applications of g, i.e., all g-applications are ground. We further need to ensure that $M \models F'(\vec{x})$ and that $M \models \forall \vec{x}.\ F'(\vec{x}) \rightarrow F(\vec{x})$. Applying Mbs on F' then ensures that $M \models F'(\vec{w})$, and therefore also $M \models F(\vec{w})$, which in turn ensures $M_0 \models F(\vec{w})$ (since M_0 agrees with M on all symbols except for $\vec{x}$). To facilitate the construction of F' as above, we first strengthen F into F_1 such that $M \models F_1$ and $\forall \vec{x}.\ F_1(\vec{x}) \rightarrow F(\vec{x})$ is valid.

Claim. Let $F_1(\vec{x})$ and $F'(\vec{x})$ be such that

- $\forall \vec{x}.\ F_1(\vec{x}) \rightarrow F(\vec{x})$ is valid,
- $M \models \forall \vec{x}.\ F'(\vec{x}) \rightarrow F_1(\vec{x})$,
- $M \models F'(\vec{x})$, and
- $M \models F'(\vec{w})$.

Then $M \models F(\vec{w})$.

We distinguish between two cases in the construction of F_1.

In the sequel, let $g(s_0), g(s_1), g(s_2), \ldots, g(s_m)$ be all the ground occurrences of g in G, F. Note that $\{s_0, \ldots, s_k\} \subseteq \{s_0, \ldots, s_n\}$.

Let $g(t_0), g(t_1), g(t_2), \dots, g(t_n)$ be all the non-ground occurrences of g in F. For each t_j we denote by lb_{t_j} and ub_{t_i} the unique terms s_i, respectively, s_{i+1}, in $\{s_1, \dots, s_k\}$ such that $M(s_i) < M(t_j) \leq M(s_{i+1})$.

[Sharon: *To get the complete instantiations for APF, should we also include s_i that do not appear in $g(s_i)$?*]

Strengthening F into F_1 with equalities. If possible, it is desirable to have Mbs use existing terms. Our first attempt is therefore to extend M_0 into a model M that not only satisfies $F(\vec{x})$, but also satisfies $\bigwedge_{j=0}^{n} \bigvee_{i=0}^{m} t_j \simeq s_i$.

To do so, if $F \wedge \bigwedge_{j=0}^{n} \bigvee_{i=0}^{m} t_j \simeq s_i$ is satisfiable, we define

$$F_1 = F \wedge \bigwedge_{j=0}^{n} \bigvee_{i=0}^{m} t_j \simeq s_i,$$

and $M = M_1 \cup M_0$ for $M_1 \models F_1^{M_0}(\vec{x})$. We have that $M \models F_1(\vec{x})$.

Strengthening F into F_1 with inequalities. If $F \wedge \bigwedge_{j=0}^{n} \bigvee_{i=0}^{m} t_j \simeq s_i$ is unsatisfiable, we consider $M = M_0 \cup M_1$ for an arbitrary model $M_1 \models F^{M_0}(\vec{x})$. In this case, we strengthen F to include range constraints on the non-ground terms according to M. This imposes restrictions on the terms generated by Mbs.

Namely, we define

$$F_1 = F \wedge \bigwedge_{j=0}^{n} lb_{t_j} < t_j \leq ub_{t_j}$$

Note that $M \models F_1(\vec{x})$.

From F_1 to F'. Our next step is to define F' where all g-applications are ground, such that Mbs may be applied, and such that M still satisfies $F'(\vec{x})$, and $M \models \forall \vec{x}.\ F'(\vec{x}) \rightarrow F_1(\vec{x})$.

To obtain F' we replace each maximal non-ground term $g(t_i)$ in F_1 by $g(s_j)$ such that $M(g(t_i)) = M(g(s_j))$, and further add all equalities $g(s_{j_1}) = g(s_{j_2})$ such that $M(g(s_{j_1})) = M(g(s_{j_2}))$. F' does not have any occurrence of $\vec{x}$ in the scope of a g-application. In addition, substituting maximal non-ground terms $g(t_i)$ ensures that no new ground terms of the form $g(s)$ are created.

[Sharon: *I think we need to record all the equalities between shared symbols that we rely on. Some may refer to terms that are not yet in the formula:*]

Example 1. Suppose $F(x) = g(g(x) + a) \simeq b$, and suppose $M \models g(x) \simeq g(c)$ and $M \models g(g(x) + a) \simeq g(d)$. Then we can either substitute $g(x) + a$ to obtain $g(d) \simeq b$ or only substitute x to obtain $g(g(c) + a) \simeq b$. The two options are not the same since it may be that $M \not\models g(c) + a \simeq d$. Maximality means that we will use $g(d) \simeq b$. Note that we also add a range constraint for x and for $g(x) + a$; the latter would be substituted by a range constraint for $g(c) + a$. The range constraint will ensure that $M \models g(g(c) + a) \simeq g(d)$.

We could also use $g(g(c)+a) \simeq b$—the claim below would still hold. However, this would introduce a new ground g-application term.

The only condition that is not immediate by this definition is that $M \models \forall \vec{x}.\ F'(\vec{x}) \to F_1(\vec{x})$.

Claim. $M \models \forall \vec{x}.\ F'(\vec{x}) \to F_1(\vec{x})$.

Proof. [Sharon: *correct? seems subtle*] The equality, respectively, inequality, constraints on non-ground terms t_i ensure that if $M, \vec{v} \models F'(\vec{x})$, then, the interpretation of $g(t_i)$ is the same as the interpretation of $g(s_j)$.

4 Summary of approach

For simplicity assume there is a single unary uninterpreted function g in the formula $F[x]$. We also assume g maps integers to integers (or reals). We claim that these simplifying assumptionsa are not material.

FixEqs extracts a set of equations that can be used to replace non-ground $g(t_j)$ by ground term $g(s_i)$, such that $M(g(t_j)) = M(g(s_i))$. The same equations can also be used to conjoin to F. For nested occurrences of non-ground g, we replace the top-most position so that substitution does not end up producing a non-ground term that cannot be handled by the substitution.

$$\text{FixEqs}(M, G_s, G_t) = \{g(t_j) = g(s_i) \mid g(t_j) \in G_t, M(g(t_j)) = M(g(s_i))\}$$

We also introduce $\mathcal{D}$ to solve disunification constraints on variables. The idea is to enumerate disunification ground candidates that are currently different than some term using head function symbol g.

$$t \sim t' = \text{Terms } t, t' \text{ are used a common relation and have the same sorts}$$

$$\mathcal{D}(g(s)) = \{t \mid M(t) \neq M(t'), t' \in \mathit{terms}(g), t \sim t'\}$$

[Sharon: *to contrast with mbqi: MBQI would instantiate $\forall x.\ \neg F(x)$ by the new instance $\neg F[x \mapsto s]$ where $M(g(x)) = M(g(s))$, without additional constraints (equalities or inequalities) on x, and without the need for an additional projection. Here, what will the new ground instance be? Suppose the skolem term generated for F_1' (of F_2') is sk, then will the instance be $\neg F[x \mapsto sk]$ or will it be $\neg F_1'[x \mapsto sk]$? (I guess you mean the former? the latter doesn't make sense, since the (in)equality constraints on t should be equivalent to true after the substitution $[x \mapsto sk]$, and the part that comes from F doesn't include x anymore since $g(t)$ was already substituted for $g(s)$.)*]

Claim. For APF, the set of terms produced from instantiation remains within a fixed set.

```
**Input:** $G \wedge \forall x \neg F[x]$, $G, F$ ground, and $M_0 \models G, M_1 \models F^{M_0}[x]$
**Data:** $g(s_0), g(s_1), g(s_2), \ldots, g(s_m)$ ground occurrences of $g$ in $G, F$
**Data:** $g(t_0), g(t_1), g(t_2), \ldots, g(t_n)$, non-ground occurrences of $g$ in $F$
**Data:** Assume $s_i = 0$ for some $i$
**Data:** Assume $M_0(s_0) < M_0(s_1) \ldots$
**Data:** Let $M = M_0 \cup M_1$
$\Theta \leftarrow \mathit{FixEqs}(M, \{g(s_0), \ldots, g(s_m)\}, \{g(t_0), \ldots, g(t_n)\})$
$F_1 \leftarrow F \wedge \bigwedge_{j=0}^{n} \bigvee_{i=0}^{m} t_j \simeq s_i$
$F_1 \leftarrow F_1 \wedge \bigwedge_{(g(s) \not\simeq x) \in F} \bigvee_{t \in \mathcal{D}(g(s))} x \simeq t$
**if** $F_1^{M_0}[x]$ *is UNSAT* **then**
    $F_1 \leftarrow F$
    **for** $j = 0$ **to** $n$ **do**
        Let $s_{j_i}, s_{j_i+1}$ be such that $M(s_{j_i}) < M(t_j) \leq M(s_{j_i+1})$
        $F_1 \leftarrow F_1 \wedge s_{j_i} < t_j \leq s_{j_{i+1}}$
    **end for**
**end if**
By construction $F_1^{M_0}$ is sat with model $M_1'$
Let $M' \leftarrow M_0 \cup M_1'$
$F_1 \leftarrow F_1[\Theta]$
$\theta \leftarrow Mbs(M', F_1)$
$G \leftarrow G \wedge \neg F[\theta]$
```

[Sharon: *If g is of sort integer to integer, I am not sure. Suppose s is a ground term in F. Suppose that after one iteration, $g(s)$ gets added. Isn't it possible that $g(s)$ gets treated as an "index" and causes a finer partition into ranges in the new interpretation of g (is there something that prevents it?), such that next time $g(g(s))$ will be added and so on?*]

Claim. For a suitable fixed set of ground $g(s_j)$; the formula F_1 is equi-satisfiable to F_2 and therefore it suffices to search for instantiations of F_1 for APF.

[Sharon: *I think you meant to start with "For APF". In general, there doesn't need to be such a fixed finite set of ground $g(s_j)$, right?*]

Claim. $F_2^{M_0}[x]$ is equisatisfiable to $F^{M_0}[x]$, where g^{M_0} is given by a set of intervals around ground occurrences $g(s_i)$. So F_2' can be used for projection and will generate a satisfiability preserving projection of F.

[Sharon: *this is because, as you wrote earlier, $F_2^{M_0}[x]$ is already satisfied by M by construction, right? this is a sanity check for me*]

Claim. Projection functions for arithmetic that select projections at bounds produce projections for F_2' that incrementally expand bounds.

This claim is intended to be useful for justifying why it suffices to use the current ground interpretation of g as a guide for instantiations. MBQI, as implemented, traverses the syntactic structure of formulas and uses the analysis to produce an instantiation set. This is useful when producing instantations of the form $t + 1$, $t - 1$, for disqualities $x \neq t$ that occur in F.

4.1 Constrained Solving

We will assume that function graphs are determined by a finite partition of input domains. For functions that take numerals as arguments it is taken to mean that the argument domain can be split into a finite set of intervals given by bounds $v_1, v_2, \ldots, v_k$, such that the function has the same value in ranges $\leq v_1, [v_1{+}1, v_2], [v_2{+}1, v_3], \ldots, v_k \leq$. Furthermore, for each interval, we associate ground witness terms for the bounds and for the range of the function. A ground witness term does not contain free quantfied variables.

4.2 Purification

[Sharon: *does it work for nested functions? e.g. $P(g(g(x) + x))$. Will we first transform it to $P(g(g(s_1){+}x)) \wedge lo_1 \leq x < hi_1 \wedge g(x) = g(s_1)$ and then transform the latter to $P(g(s_2)) \wedge lo_2 \leq g(s_1) + x < hi_2 \wedge g(g(s_1) + x)) = g(s_2) \wedge lo_1 \leq x < hi_1 \wedge g(x) = g(s_1)$? (we can do it from the outside in, and it will give the same result)*
Another option is to first normalize by introducing auxiliary variables $P(g(y)) \wedge y = g(x) + x$ or even $P(g(y)) \wedge y = z + x \wedge z = g(x)$. Applying the transformation on the normalized form will yield $P(g(s_2)) \wedge lo_2 \leq y < hi_2 \wedge g(s_2) = g(y) \wedge y = z + x \wedge z = g(s_1) \wedge lo_1 \leq x < hi_1 \wedge g(x) = g(s_1)$. Then let Mbs eliminate the auxiliary variables as well. This seems unnecessary. It will generate skolem terms not only for x but also for y which stands for $g(x)+x$] [Nikolaj: *I think we should consider two cases: nested functions and domain restricted formulas. With domain restriction it should be a property that variables are solved using linear combinations of ground terms. In the general case, where domain restriction is too strong, we solve first ignoring subterms as much as possible, and then substitute values for variables if the resulting projection fails to solve for a variable.*].

4.3 Expanding the candidate of ground instantiations

As the coin-example shows, surjectivity is not well covered by the instantiation strategy that looks for only terms under a function g.

```
A x y : g(h(x, y), y) = x
```

Elsewhere in F there are ground terms g, but h only occurs in this quantifier. Recall that the goal is to find terms for

```
E x y : g(h(x, y), y) != x
```

So instances for x that are different than ground occurrences of $g(s)$ may be admissible. A first heuristic could be to mine for terms from the original ground portion, G, that have the same type as x, but are different than $g(s)$. Thus, for each ground $g(s)$ enumerate terms t_j such that $M \models g(s) \neq t_j$. Syntactically, choosing the terms from atomic formulas where $g(s)$ occurs may narrow the candidates.

5 Coin Transfer Example

$$
\begin{array}{rl}
(A1) & \mathit{sum}' > \mathit{sum} \text{ (other case is symmetric)} \\
(A2) & \mathit{bal}'(a_0) = \mathit{bal}(a_0) + 1 \\
(A3) & \mathit{bal}'(a_1) = \mathit{bal}(a_1) - 1 \\
(A4) & A \neq a_0 \wedge A \neq a_1 \rightarrow \mathit{bal}'(A) = \mathit{bal}(A) \\
(IS) & N > 0 \rightarrow \mathit{ind}(\mathrm{na2c}(N, A), A) = N \\
(BS) & \mathit{ind}(C, A) \leq \mathit{bal}'(A) \rightarrow \mathit{count}(C) \leq \mathit{sum}' \\
(SB) & \mathit{ind}(C, \mathrm{c2a}(C)) \leq \mathit{bal}'(\mathrm{c2a}(C)) \leftarrow \mathit{count}(C) \leq \mathit{sum}' \\
(\mathit{disj}) & \mathit{ind}(C, A_1) \leq \mathit{bal}(A_1) \wedge \mathit{ind}(C, A_2) \leq \mathit{bal}(A_2) \rightarrow A_1 = A_2 \\
(CS) & N > 0 \rightarrow \mathit{count}(\mathrm{n2c}(N)) = N \\
(\mathit{inj}) & \mathit{ind}(C_1, A) = \mathit{ind}(C_2, A) \rightarrow C_1 = C_2
\end{array}
$$

Proof:

let $c_1 = \text{na2c}(\mathit{bal}(a_1), a_1)$	
(A3)	$\mathit{bal}(a_1) > 0$
inst. ISon $\mathit{bal}(a_1), a_1$:	$\mathit{ind}(c_1, a_1) = \mathit{bal}(a_1) > \mathit{bal}'(a_1)$
inst. BS(bal) on c_1, a_1:	$\mathit{count}(c_1) \leq \mathit{sum} < \mathit{sum}'$
inst. SB(bal') on c_1:	$\mathit{ind}(c_1, \text{c2a}(c_1)) \leq \mathit{bal}'(\text{c2a}(c_1))$
$\mathit{ind}(c_1, a_1) > \mathit{bal}'(a_1) \implies$	$\text{c2a}(c_1) \neq a_1$
inst. disj on $c_1, a_1, \text{c2a}(c_1) \implies$	$\mathit{ind}(c_1, \text{c2a}(c_1)) > \mathit{bal}(\text{c2a}(c_1))$
(A4)	$\text{c2a}(c_1) \neq a_0 \to \mathit{bal}(\text{c2a}(c_1)) = \mathit{bal}'(\text{c2a}(c_1))$
$\implies$	$\text{c2a}(c_1) = a_0$
$\implies$	$\mathit{ind}(c_1, a_0) \leq \mathit{bal}'(a_0)$
	$\mathit{ind}(c_1, a_0) > \mathit{bal}(a_0)$
(A2) $\implies$	$\mathit{ind}(c_1, a_0) = \mathit{bal}(a_0) + 1 = \mathit{bal}'(a_0)$
let $c_2 = \text{n2c}(\mathit{sum}')$	
(A1)	$\mathit{sum}' > 0$
inst. CSon sum'	$\mathit{count}(c_2) = \mathit{sum}' > \mathit{sum}$
inst. SB(bal') on c_2:	$\mathit{ind}(c_2, \text{c2a}(c_2)) \leq \mathit{bal}'(\text{c2a}(c_2))$
inst. BS(bal) on $c_2, \text{c2a}(c_2)$:	$\mathit{ind}(c_2, \text{c2a}(c_2)) > \mathit{bal}(\text{c2a}(c_2))$
(A4+A3) $\implies$	$\text{c2a}(c_2) = a_0$
$\implies$	$\mathit{ind}(c_2, a_0) = \mathit{bal}'(a_0)$
inst. inj on $c_1, c_2, a_0 \implies$	$c_1 = c_2$
$\implies$	$\mathit{count}(c_1) = \mathit{sum}'$
$\implies$	$\perp$

Hints for the instantiations (rough idea): Let f stand for *ind*, g for *count*, h for *bal*, and c for *sum*.

Then BS:

$$\forall x, y.\ f(x, y) \leq h(y) \to g(x) \leq c$$

$f(x, y) \leq h(y)$ as a precondition indicates that we may want to consider the f-pre-image of $h(y)$, i.e., indicates that we want to instantiate surjectivity of f on $h(y)$, where the terms relevant for y are obtained as usual.

SB:

$$\forall x.\ g(x) \leq c \to f(x, r(x)) \leq h(x)$$

$g(x) \leq c$ as a precondition indicates that we may want to consider $g^{-1}(c)$ for x, i.e., indicates that we want to instantiate surjectivity of g on c.

More generally, suppose g is a function axiomatized to be surjective: $\forall z, y.\ \exists x.\ g(x, z) = y$. If $g(x, t_1) \bowtie t_2$ appears as a guard in a formula, where $\bowtie \in \{\leq, \geq, =\}$, then instantiate surjectivity of g on (t_1, t_2).

Specifically, suppose $F = F' \wedge \varphi$, where $\varphi = \forall x, y.\ (g(x, t_1(y)) \bowtie t_2(y)) \to \psi(x)$. Suppose $M \models \varphi$ and $M \models \exists x.\ \neg\psi(x)$. It follows that $M \models \exists x.\ \neg\psi(x) \wedge \forall y.\ \neg(g(x, t_1(y)) \bowtie t_2(y))$. Then instantiate surjectivity of g on $(t_1(w), t_2(w))$, where w ranges over instantiations of y computed as usual[Sharon: *probably can do better here*].

Example 2. BS:

$$\varphi = \forall C, A.\ \mathit{ind}(C, A) \leq \mathit{bal}(A) \rightarrow \mathit{count}(C) \leq \mathit{sum}$$

$$\psi = \mathit{count}(C) \leq \mathit{sum}$$

Without the surjectivity axiom for *ind*, the following model is obtained:

$$\begin{aligned}
D(\mathit{Coin}) &= \{c_0, c_1, c_2\} \\
D(\mathit{Address}) &= \{ad_0, ad_1, ad_2, ad_3\} \\
M(a_0) &= ad_1 \\
M(a_1) &= ad_0 \\
M(\mathit{sum}') &= 3 \\
M(\mathit{sum}) &= 2 \\
M(\mathit{count}) &= \{c_2 \mapsto 2, c_0 \mapsto 3, c_1 \mapsto 1, \_ \mapsto 45\} \\
M(\mathit{bal}') &= \{ad_3 \mapsto 1, ad_2 \mapsto 5, ad_1 \mapsto 1, ad_0 \mapsto 3, \_ \mapsto 46\} \\
M(\mathit{bal}) &= \{ad_3 \mapsto 1, ad_2 \mapsto 5, ad_1 \mapsto 0, ad_0 \mapsto 4, \_ \mapsto 48\}
\end{aligned}$$

$$\begin{aligned}
M(\mathit{ind}) = \{&(c_2, ad_1) \mapsto 2, \\
&(c_2, ad_3) \mapsto 1, \\
&(c_0, ad_3) \mapsto 2, \\
&(c_0, ad_2) \mapsto 6, \\
&(c_0, ad_0) \mapsto 7, \\
&(c_0, ad_1) \mapsto 1, \\
&(c_1, ad_1) \mapsto 3, \\
&(c_1, ad_0) \mapsto 8, \\
&(c_1, ad_2) \mapsto 1, \\
&\_ \mapsto 47\}
\end{aligned}$$

$M \models \exists C.\ \mathit{count}(C) > \mathit{sum} \wedge \forall A.\ \mathit{ind}(C, A) > \mathit{bal}(A)$. The witness is $v(C) = c_0$.

Therefore, instantiate the surjectivity axiom of *ind* on $(s, \mathit{bal}(s))$ where s is any instantiation associated with A (i.e., associated with the argument of *bal*). In our case, this includes a_1 as desired, generating the instantiation $(a_1, \mathit{bal}(a_1))$ (and also $(a_0, \mathit{bal}(a_0))$).

Axioms

```
(set-logic UFLIA)
(declare-sort Coin 0)
(declare-sort Address 0)
(declare-fun old-sum () Int)
(declare-fun new-sum () Int)
```

```
(declare-fun a0 () Address)
(declare-fun a1 () Address)
(declare-fun old-bal (Address) Int)
(declare-fun new-bal (Address) Int)
(declare-fun count (Coin) Int)
(declare-fun ind (Coin Address) Int)
(declare-fun na2c (Int Address) Coin)
(declare-fun n2c (Int) Coin)
(declare-fun c2a (Coin) Address)

(assert (<= 0 old-sum))
(assert (<= 0 new-sum))
(assert (= (new-bal a0) (+ (old-bal a0) 1)))
(assert (= (old-bal a1) (+ (new-bal a1) 1)))

(assert (forall ((C Coin) (D Coin) (A Address) (B Address) (N Int))
 (and
  (< 0 (count C))
  (=> (= (count C) (count D)) (= C D))
  (<= 0 (old-bal A))
  (<= 0 (new-bal A))
  (< 0 (ind C A))
  (=> (= (ind C A) (ind D A)) (= C D))
; (=> (< 0 N) (= (ind (na2c N A) A) N))
; (=> (< 0 N) (= (count (n2c N)) N))
  (=> (<= (ind C A) (old-bal A)) (<= (count C) old-sum))
  (=> (<= (count C) old-sum) (<= (ind C (c2a C)) (old-bal (c2a C))))
  (=> (<= (ind C A) (new-bal A)) (<= (count C) new-sum))
  (=> (<= (count C) new-sum) (<= (ind C (c2a C)) (new-bal (c2a C))))
  (=> (and (<= (ind C A) (old-bal A)) (<= (ind C B) (old-bal B))) (= A B))
  (=> (and (<= (ind C A) (new-bal A)) (<= (ind C B) (new-bal B))) (= A B))
  (=>  (and (distinct A a0) (distinct A a1)) (= (old-bal A) (new-bal A)))
  (distinct old-sum new-sum))))
```

Model without surjectivity axioms:

```
(declare-fun a0 () Address)
(declare-fun a1 () Address)
(forall ((x Address)) (or (= x a0) (= x a1)))
(declare-fun c0 () Coin)
(forall ((x Coin)) (= x c0))

(define-fun a0 () Address a0)
(define-fun new-sum () Int 1)
(define-fun a1 () Address a1)
(define-fun old-sum () Int 0)
(define-fun c2a ((x Coin)) Address a1)
(define-fun count ((x Coin)) Int 5)
(define-fun na2c ((x Int) (y Address)) Coin c0)
```

```
  (define-fun n2c ((x Int)) Coin c0)
  (define-fun new-bal ((x Address)) Int (ite (= x a1) 0 1))
  (define-fun ind ((x Coin) (y Address)) Int 7)
  (define-fun old-bal ((x Address)) Int (ite (= x a1) 1 0))
```

Instantiation:

$$N > 0 \Rightarrow count(n2c(N)) = N$$

where n2c$^M := \lambda N\ .c0\ count(c0)^M = 5$.

```
solve for

c0 = n2c(N) and count(c0) != N and N > 0

reduces to

count(c0) != N and N > 0

and projects to

N = count(c0) + 1 and count(c0) >= 0

But projecting to count(c0) + 1 or count(c0) - 1 or 1 is
not going to be sufficient. What we would like simulate
is an inference:

  (N = count(n2c(N)) or N <= 0) & count(c0) > bal(a0)

or similar.

The unification constraints that eliminate the equality are:

  bal(a0) = N, c0 = n2c(N), count(c0) != bal(a0)

Similar, for

  (N = ind(na2c(N,A),A) = N or N <= 0) & inc(c0, a0) > bal(a0)

we have unification constraints

  bal(a0) = N, c0 = na2c(N,A), a0 = A, inc(c0, a0) != bal(a0)
```

The unification constraints

$$c0 = n2c(N),$$
$$c0 = na2c(N, A)$$

are resolved by the model M, where $\text{na2c}^M = \lambda(N, A).c_0$ and $\text{n2c}^M = \lambda N.c_0$. The other unification constraints give solutions to the bound variables.

Note that neither FixEqs unification constrains, nor bounds constraints on domains are sufficient to get these unification constraints. In the following we propose an instantiation rule that specifically targets the case from our example. A candidate inference rule is as follows:

$$\frac{(X \simeq t \vee C) \qquad M(t') \neq M(s') \qquad M(t\theta) = M(t') \qquad M(s') = M(X\theta)}{(X \simeq t \vee C)\theta}$$

where C is a clause, t is a term that main contain variables, t', s' ground terms different in the current model M, and θ is a solution to bound variables that sets X to s' and t to t'. The side-condition that s', t' are different in the current model M, ensures that the inference either blocks M or the equation $X = t$ under M. The examples suggest we can implement the inference rule as follows:

1. Consider the clause $X \simeq g(t) \vee C$.
2. Let M be a model of current ground literals.
3. Let S be the set of s', such that there is $g(t')$ with $M(g(t')) \neq M(s')$.
4. Add the difference constraint $X \in S$, which expands into $\bigvee_{s \in S} X \simeq s$, to the unification constraints for the two versions of F_1. The difference constraints restrict solutions of X to be based on terms already in the ground portion G.

6 Open Problems

- Extend projection to work with mix of algebraic data-types and arithmetic. For example $cons(x + 1, \ell)$, is an ADT term that uses $x + 1$ in the head. If x is a bound variable, then x occurs under an interpreted function, but from a different theory. Colmerauer and student developed quantifier elimination for such mixes of arithmetic and ADTs, so a suitable fragment can be expected to be decidable. Is there a good way to perform projection for such mixed terms?
- Would extending models for g to be affine functions instead of piecewise constant functions produce anything useful?
- What other combinations of infinite models/induction make sense. Taking an example from Teucke et al; $\forall x.R(x, x)$, $\forall x, y.R(g(x), g(y)) \implies R(x, y)$, $\forall x.\neg R(c, g(x))$ is satisfiable where R is an equality relation and $c \neq g(x)$. Their approach is not through interpreting R as an equality relation. They split R into a reflexive and irreflexive part and produce a set of clauses that imply preservation of satisfiability for the original clause set.

7 Infinite Models

Consider a function symbol $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \rightarrow \mathsf{s}$, where s is an uninterpreted sort. We define a template space for interpretations of f, and, accordingly, for the universe associated with sort s. We first define *base interpretations*:

- **A Herbrand function $\mathbf{f}_H$** defined by $\mathbf{f}_H(e_1, \ldots, e_n) = f(t_1, \ldots, t_n)$ for some terms $t_1, \ldots, t_n$ such that $t_i^M = e_i$.
- **A selection function $\mathbf{f}_{g,i}$** defined by $\mathbf{f}_{g,i}(e) = t_i$ for some function symbol g and terms $t_1, \ldots, t_n$ such that $g(t_1, \ldots, t_n)^M = e$. A selection function is suitable when f has a single argument.
- **A projection function $\mathbf{f}_i$** defined by $\mathbf{f}_i(e_1, \ldots, e_n) = e_i$ for some i. [Sharon: *alternatively, can map to a term t_i such that $t_i^M = e_i$ here as well – what do we want the universe of* s *to consist of?*]
- **A constant functions $\mathbf{f}_e$** defined by $\mathbf{f}_e(e_1, \ldots, e_n) = e$ for some element e in the universe of s.

This means that the universe associated with s consists of terms $f(t_1, \ldots, t_n)$ (in the case of Herbrand functions) as well as t_i (in the case of selection functions).

The space of interpretations of f consists of finite combinations of base interpretations.

What if s *is interpreted? (e.g., int)* Consider the surjectivity axiom from the example

$$N > 0 \Rightarrow \mathit{count}(n2c(N)) = N$$

N is of sort int. In this case, an interpretation that would satisfy the axiom is:

$$\mathbf{n2c}(n) = n2c(n) \qquad \mathbf{count}(n2c(n)) = n$$

We use a Herbrand function for $n2c$, and a selection function for count.