1 Introduction

Reasoning about probrams bla bla bla

2 Programming Language

2.1 Syntax

We start by defining an imperative programming language with non deterministic choice and parametric on a set *Base* of base commands, common choices for the set of base commands usually contain a command for assignments and boolean guards.

The set of valid $\mathbb{C}$ programs is defined by the following inductive definition:

Definition 1 ($\mathbb{C}$ language syntax)

$$\begin{aligned}
\mathbb{C} ::=\ & \mathbb{0} && \textit{Always non terminating program} \\
|\ & \mathbb{1} && \textit{Identity program (skip)} \\
|\ & b && \textit{Base command} \\
|\ & C_1 \fatsemi C_2 && \textit{Program composition} \\
|\ & C_1 + C_2 && \textit{Non deterministic choice} \\
|\ & C^\star && \textit{Iteration}
\end{aligned}$$

Where $C, C_1, C_2 \in \mathbb{C}$ and $b \in Base$.

2.2 Semantics

Given a set $\mathbb{S}$ of states and a family of partial functions $[\![b]\!] : \mathbb{S} \hookrightarrow \mathbb{S}$ we can define inductively the semantics of a program in $\mathbb{C}$ by structural induction on the terms:

Definition 2 ($\mathbb{C}$ language semantics)

$$\begin{aligned}
[\![\cdot]\!]\ &:\ \mathcal{P}(\mathbb{S}) \to \mathcal{P}(\mathbb{S}) \\
[\![\mathbb{0}]\!] &= const\ \emptyset \\
[\![\mathbb{1}]\!] &= id \\
[\![b]\!] &= \lambda P \to \{x \mid [\![b]\!](p) \downarrow = x\ \wedge p\ \in P\} \\
[\![C_1 \fatsemi C_2]\!] &= [\![C_2]\!] \circ [\![C_1]\!] \\
[\![C_1 + C_2]\!] &= \lambda P \to [\![C_1]\!]P \cup [\![C_2]\!]P \\
[\![C^\star]\!] &= \lambda P \to lfp(\lambda P' \to P \cup [\![C]\!]P')
\end{aligned}$$

Theorem 1 ($[\![\cdot]\!]$ is monotone) $\forall P, Q \in \mathcal{P}(\mathbb{S}) \quad C \in \mathbb{C}$

$$P \subseteq Q \implies [\![C]\!](P) \subseteq [\![C]\!](Q)$$

This framework is general enough to cover non deterministic imperative languages, for example if we include a base command for boolean guards $e?$ where e is a boolean valued expression on $\mathbb{S}$ we can define the usual control flow statements `if` b `then` C_1 `else` C_2 as $(e? \fatsemi C_1) + (\neg e? \fatsemi C_2)$ and `while` e `do` C `done` as $(e? \fatsemi C)^\star \fatsemi \neg e?$.

3 Best Abstract Inductive semantics

We can abstract the *Best Abstract Inductive Semantics* of a program in $\mathbb{C}$ parametrically on a complete lattice A that has as elements collections of program states and a family of monotone functions $[\![b]\!]^A_{bai} : A \to A \quad \forall\ b \in \mathit{Base}$ by structural induction on the syntax of $\mathbb{C}$:

Definition 3 (Best abstract inductive semantics)

$$\begin{aligned}
[\![\cdot]\!]^A_{bai} &: A \to A \\
[\![\mathbb{0}]\!]^A_{bai} &= \mathit{const}\ \bot_A \\
[\![\mathbb{1}]\!]^A_{bai} &= \mathit{id}_A \\
[\![b]\!]^A_{bai} &= \lambda P \to [\![b]\!]^A_{bai}(P) \\
[\![C_1 \fatsemi C_2]\!]^A_{bai} &= [\![C_2]\!]^A_{bai} \circ [\![C_1]\!]^A_{bai} \\
[\![C_1 + C_2]\!]^A_{bai} &= \lambda P \to [\![C_1]\!]^A_{bai}(P) \vee_A [\![C_2]\!]^A_{bai}(P) \\
[\![C^\star]\!]^A_{bai} &= \lambda P \to lfp(\lambda P' \to P \vee_A [\![C]\!]^A_{bai} P')
\end{aligned}$$

Theorem 2 (Semantic equivalence) *If we take as the lattice $\mathcal{P}(\mathbb{S})$ and as $[\![b]\!]^A_{bai} = \lambda P \to \{x \mid [\![b]\!](p) \downarrow = x \ \wedge \ p \in P\}$ the two semantics are identical.*
$\forall P \in \mathcal{P}(\mathbb{S}) \quad C \in \mathbb{C}$

$$[\![C]\!]^{\mathcal{P}(\mathbb{S})}_{bai}(P) = [\![C]\!](P)$$

Theorem 3 ($[\![\cdot]\!]^A_{bai}$ is monotone) $\forall P, Q \in A \quad C \in \mathbb{C}$

$$P \leq_A Q \implies [\![C]\!]^A_{bai}(P) \leq_A [\![C]\!]^A_{bai}(Q)$$

3.1 Obtaining BAIs from other BAIs via Galois connections

Given a Galois connection $\langle D, \leq_D \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$ if we have define a *Best Abstract Inductive Semantics* on D with the semantics of basic commands $[\![b]\!]^D_{bai}$ we can define a *Best Abstract Inductive Semantics* on A with semantics of basic commands $[\![b]\!]^A_{bai} = \alpha \circ [\![b]\!]^D_{bai} \circ \gamma$.

Theorem 4 (Soundness) $\forall P \in D\ C \in \mathbb{C}$

$$\alpha([\![C]\!]^D_{bai}(P) \leq_D [\![C]\!]^A_{bai}(\alpha(P))$$

4 Abstract Hoare logic

In this section we will define when an Abstract Hoare triple is valid, give some inference rules for it BLAH.

Definition 4 (Abstract Hoare triple) *Fixed a complete lattice A and the semantics of the base commands $[\![b]\!]^A_{bai}$, an Abstract Hoare triple is valid if and only if executing the bai of a command C of some precondition captured by the element P of A is overapproximated by some element Q of A:*

$$\langle P \rangle_A \; C \; \langle Q \rangle \iff [\![C]\!]^A_{bai}(P) \leq_A Q$$

4.1 Inference rule

As in standard Hoare logic we can give a set of inference rules to derive valid triples from smaller ones:

Definition 5 (Abstract Hoare inference rules) $\dfrac{}{\vdash \langle P \rangle_A \; \mathbb{0} \; \langle \bot \rangle} \; (\mathbb{0})$

$$\dfrac{}{\vdash \langle P \rangle_A \; \mathbb{1} \; \langle P \rangle} \; (\mathbb{1})$$

$$\dfrac{}{\vdash \langle P \rangle_A \; b \; \langle [\![b]\!]^A_{bai}(P) \rangle} \; (b)$$

$$\dfrac{\vdash \langle P \rangle_A \; C_1 \; \langle Q \rangle \qquad \vdash \langle Q \rangle_A \; C_2 \; \langle R \rangle}{\vdash \langle P \rangle_A \; C_1 \fatsemi C_2 \; \langle R \rangle} \; (\fatsemi)$$

$$\dfrac{\vdash \langle P \rangle_A \; C_1 \; \langle Q \rangle \qquad \vdash \langle P \rangle_A \; C_2 \; \langle Q \rangle}{\vdash \langle P \rangle_A \; C_1 + C_2 \; \langle Q \rangle} \; (+)$$

$$\dfrac{\vdash \langle P \rangle_A \; C \; \langle P \rangle}{\vdash \langle P \rangle_A \; C^* \; \langle P \rangle} \; (*)$$

$$\dfrac{P \leq P' \qquad \vdash \langle P' \rangle_A \; C \; \langle Q' \rangle \qquad Q' \leq Q}{\vdash \langle P \rangle_A \; C \; \langle Q \rangle} \; (\leq)$$

Example 4.1 (Derivation in the integer interval domain) *We will start by defining our running example the integer interval domain: The elements of the interval domain on one variable are defined by $Int = \{[x, y] \mid x \leq y \quad x, y \in \mathbb{Z} \cup \{+\infty, -\infty\}\} \cup \{\bot\}$ where the order relation is given by $\bot \leq_{Int} a \; \forall a \in Int$ and $[a, b] \leq_{Int} [c, d] \iff c \leq a \wedge b \leq d$ and clearly $\top = [-\infty, +\infty]$. The definition can be lifted pointwise for domains with an arbitrary number of variables.*

As basic command we will add $e?$ for boolean tests discarding all the states that don't satisfy the condition e and $x := y$ assigning the result of evaluating expression y to the variable x and fix $[\![b]\!]^A_{bai} = \alpha \circ [\![b]\!] \circ \gamma$.

Then the following is a valid derivation for program $C = (x := 1 + x := 2) \fatsemi x := x + 1$

$$\dfrac{\dfrac{\dfrac{\top \leq \top \qquad \vdash \langle \top \rangle_A\ x := 1\ \langle x \in [1,1] \rangle \qquad x \in [1,1] \leq x \in [1,2]}{\vdash \langle \top \rangle_A\ x := 1\ \langle x \in [1,2] \rangle} \qquad \dfrac{\top \leq \top \qquad \vdash \langle \top \rangle_A\ x := 2\ \langle x \in [2,2] \rangle \qquad x \in [2,2] \leq x \in [1,2]}{\vdash \langle \top \rangle_A\ x := 2\ \langle x \in [1,2] \rangle}}{\vdash \langle \top \rangle_A\ x := 1 + x := 2\ \langle x \in [1,2] \rangle} \qquad \vdash \langle x \in [1,2] \rangle_A\ x := x + 1\ \langle x \in [2,3] \rangle}{\vdash \langle \top \rangle_A\ C\ \langle x \in [2,3] \rangle}$$

And clearly $[\![C]\!]^A_{bai}(\top) = x \in [2,3]$.

Theorem 5 (The proofsystem is sound)

$$\vdash \langle P \rangle_A\ C\ \langle Q \rangle \implies \langle P \rangle_A\ C\ \langle Q \rangle$$

Proof 1 *By structural induction on the last rule applied in the derivation of* $\vdash \langle P \rangle_A\ C\ \langle Q \rangle$*:*

- $(\mathbb{0})$*: Then the last step in the derivation was:*

  $$\dfrac{}{\vdash \langle P \rangle_A\ \mathbb{0}\ \langle \bot \rangle}\ (\mathbb{0})$$

  The triple is valid since:

  $$[\![\mathbb{0}]\!]^A_{bai}(P) = \bot_A \qquad \textit{By definition of } [\![\cdot]\!]^A_{bai}$$

- $(\mathbb{1})$*: Then the last step in the derivation was:*

  $$\dfrac{}{\vdash \langle P \rangle_A\ \mathbb{1}\ \langle P \rangle}\ (\mathbb{1})$$

  The triple is valid since:

  $$[\![\mathbb{1}]\!]^A_{bai}(P) = P \qquad \textit{By definition of } [\![\cdot]\!]^A_{bai}$$

- (b)*: Then the last step in the derivation was:*

  $$\dfrac{}{\vdash \langle P \rangle_A\ b\ \langle [\![b]\!]^A_{bai}(P) \rangle}\ (b)$$

  The triple is valid since:

  $$[\![b]\!]^A_{bai}(P) = [\![b]\!]^A_{bai}(P) \qquad \textit{By definition of } [\![\cdot]\!]^A_{bai}$$

- $(\fatsemi)$*: Then the last step in the derivation was:*

  $$\dfrac{\vdash \langle P \rangle_A\ C_1\ \langle Q \rangle \qquad \vdash \langle Q \rangle_A\ C_2\ \langle R \rangle}{\vdash \langle P \rangle_A\ C_1 \fatsemi C_2\ \langle R \rangle}\ (\fatsemi)$$

  By inductive hypothesis: $[\![C_1]\!]^A_{bai}(P) \leq_A Q$ *and* $[\![C_2]\!]^A_{bai}(Q) \leq_A R$*.*

  The triple is valid since:

  $$\begin{aligned} [\![C_1 \fatsemi C_2]\!]^A_{bai}(P) &= [\![C_2]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P)) && \textit{By definition of } [\![\cdot]\!]^A_{bai} \\ &\leq_A [\![C_2]\!]^A_{bai}(Q) && \textit{By monotonicity of } [\![\cdot]\!]^A_{bai} \\ &\leq_A R \end{aligned}$$

- $(+)$*: Then the last step in the derivation was:*

$$\frac{\vdash \langle P\rangle_A\ C_1\ \langle Q\rangle \qquad \vdash \langle P\rangle_A\ C_2\ \langle Q\rangle}{\vdash \langle P\rangle_A\ C_1 + C_2\ \langle Q\rangle}\ (+)$$

By inductive hypothesis: $[\![C_1]\!]^A_{bai}(P) \leq Q$ *and* $[\![C_2]\!]^A_{bai}(P) \leq Q$.

The triple is valid since:

$$\begin{aligned}[\![C_1 + C_2]\!]^A_{bai}(P) &= [\![C_1]\!]^A_{bai}(P) \vee [\![C_2]\!]^A_{bai}(P) && \textit{By definition of } [\![\cdot]\!]^A_{bai}\\ &\leq_A Q \vee Q\\ &= Q\end{aligned}$$

- $(\star)$*: Then the last step in the derivation was:*

$$\frac{\vdash \langle P\rangle_A\ C\ \langle P\rangle}{\vdash \langle P\rangle_A\ C^\star\ \langle P\rangle}\ (*)$$

By inductive hypothesis: $[\![C]\!]^A_{bai}P \leq P$

$$[\![C^\star]\!]^A_{bai}(P) = lfp(\lambda P' \to P \vee_A [\![C]\!]^A_{bai}(P'))$$

$$\begin{aligned}(\lambda P' \to P \vee_A [\![C]\!]^A_{bai}(P'))(P) &= P \vee [\![C]\!]^A_{bai}(P) && \textit{since } [\![C]\!]^A_{bai}(P) \leq P\\ &= P\end{aligned}$$

Hence P *is a fixpoint for* $\lambda P' \to P \vee_A [\![C]\!]^A_{bai}(P')$

Thus $lfp(\lambda P' \to P \vee_A [\![C]\!]^A_{bai}(P')) \leq_A P$

- $(\leq)$*: Then the last step in the derivation was:*

$$\frac{P \leq P' \qquad \vdash \langle P'\rangle_A\ C\ \langle Q'\rangle \qquad Q' \leq Q}{\vdash \langle P\rangle_A\ C\ \langle Q\rangle}\ (\leq)$$

By inductive hypothesis: $[\![C]\!]^A_{bai}(P') \leq Q'$.

$$\begin{aligned}[\![C]\!]^A_{bai}(P)[\![C]\!]^A_{bai}(P') & && \textit{By monotonicity of } [\![\cdot]\!]^A_{bai}\\ &\leq Q' && \textit{By inductive hypothesis}\\ &\leq Q\end{aligned}$$

4.2 Merge rules

In standard Hoare loigc the following rule is valid:

$$\frac{\vdash \{P_1\}\ C\ \{Q\} \qquad \vdash \{P_2\}\ C\ \{Q\}}{\vdash \{P_1 \vee P_2\}\ C\ \{Q\}}\ (merge)$$

And can be used to easily merge the results obtained by different analysis, but in general the same rule in the context of abstract Hoare logic:

$$\frac{\vdash \langle P_1\rangle_A\ C\ \langle Q\rangle \qquad \vdash \langle P_2\rangle_A\ C\ \langle Q\rangle}{\vdash \langle P_1 \vee P_2\rangle_A\ C\ \langle Q\rangle}\ (merge)$$

is unsound.

Example 4.2 (Unsoundness of the rule $(merge)$**)** *Let A and $[\![b]\!]^A_{bai}$ be the same as in 4.1 and $C = (x = 3?\ ⨾\ x := 400) + (x \neq 3?\ ⨾\ x := x + 1)$*

Then we can derive the following triples:

- $\vdash \langle x \in [1, 2]\rangle_A\ C\ \langle x \in [2, 3]\rangle$
- $\vdash \langle x \in [4, 5]\rangle_A\ C\ \langle x \in [5, 6]\rangle$

But applying the rule $(merge)$ would allow to derive the following: $\vdash \langle x \in [1, 5]\rangle_A\ C\ \langle x \in [2, 6]\rangle$, that is unsound since $[\![C]\!]^A_{bai}(x \in [1, 5]) = x \in [2, 400]$.

The cause of the issue is caused by the non additivity of the base commands, in fact:

Theorem 6 (Additivity of the semantics on attitive base commands) *if $\forall P_1, P_2 \in A$ and $b \in Base$ $[\![B]\!]^A_{bai}(P_1 \vee P_2) = [\![B]\!]^A_{bai}(P_1) \vee [\![P_2]\!]^A_{bai}$ then: $\forall P_1, P_2 \in A \quad C \in \mathbb{C}$*

$$[\![C]\!]^A_{bai}(P_1) \vee [\![C]\!]^A_{bai}(P_2) = [\![C]\!]^A_{bai}(P_1 \vee P_2)$$

Proof 2 *By structural induction on C:*

- $\mathbb{0}$*: By definition $[\![\mathbb{0}]\!]^A_{bai}(P_1 \vee P_2) = \bot$ and $[\![\mathbb{0}]\!]^A_{bai}(P_i) = \bot$*
- $\mathbb{1}$*: By definition $i \in \{1, 2\}$ $[\![\mathbb{1}]\!]^A_{bai}(P_i) = P_i$ and $[\![\mathbb{1}]\!]^A_{bai}(P_1 \vee P_2) = P_1 \vee P_2$*
- *b:*

$$[\![b]\!]^A_{bai}(P_1 \vee P_2) = [\![b]\!]^A_{bai}(P_1) \vee [\![b]\!]^A_{bai}(P_2) \qquad \textit{By additivity of } [\![b]\!]^A_{bai}$$

- *$C_1 ⨾ C_2$:*

$$\begin{aligned}
[\![C_1 ⨾ C_2]\!]^A_{bai}(P_1 \vee P_2) &= [\![C_2]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P_1 \vee P_2)) \\
&= [\![C_2]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P_1) \vee [\![C_1]\!]^A_{bai}(P_2)) \quad \textit{By inductive hypothesis} \\
&= [\![C_1 ⨾ C_2]\!]^A_{bai}(P_1) \vee [\![C_2 ⨾ C_2]\!]^A_{bai}(P_2) \quad \textit{By inductive hypothesis}
\end{aligned}$$

- $C_1 + C_2$:

$$\begin{aligned}
[\![C_1 + C_2]\!]^A_{bai}(P_1 \vee P_2) &= [\![C_1]\!]^A_{bai}(P_1 \vee P_2) \vee [\![C_2]\!]^A_{bai}(P_1 \vee P_2) \\
&= [\![C_1]\!]^A_{bai}(P_1) \vee [\![C_2]\!]^A_{bai}(P_2) \\
&\vee [\![C_2]\!]^A_{bai}(P_1) \vee [\![C_2]\!]^A_{bai}(P_2) && \textit{By inductive hypothesis} \\
&= [\![C_1 + C_2]\!]^A_{bai}(P_1) \vee [\![C_1 + C_2]\!]^A_{bai}(P_2) && \textit{Rearranging } \vee
\end{aligned}$$

- $C^\star$:

$$[\![C^\star]\!]^A_{bai}(P_1 \vee P_2) = lfp(\lambda P' \to P_1 \vee P_2 \vee [\![C]\!]^A_{bai}(P'))$$

Let $F_i = lfp(\lambda P' \to P_i \vee [\![C]\!]^A_{bai}(P'))$

$$\begin{aligned}
(\lambda P' \to P_1 \vee P_2 \vee_A [\![C]\!]^A_{bai}(P'))(F_1 \vee F_2) &= P_1 \vee P_2 \vee [\![C]\!]^A_{bai}(F_1 \vee F_2) \\
&\textit{By inductive hypothesis} \\
&= P_1 \vee P_2 \vee [\![C]\!]^A_{bai}(F_1) \vee [\![C]\!]^A_{bai}(F_2) \\
&= P_1 \vee [\![C]\!]^A_{bai}(F_1) \vee P_2 \vee [\![C]\!]^A_{bai}(F_2) \\
&= F_1 \vee F_2
\end{aligned}$$

Thus $F_1 \vee F_2$ *is a fixpoint for* $\lambda P' \to P_i \vee [\![C]\!]^A_{bai}(P')$.

Following the same reasoning is also the least one by $F_1 \vee F_2$ *beeing the smallest* $K \geq F_1$ *and* $K \geq F_2$

Theorem 7 (Soundness of the rule *merge* on attitive base commands) *if* $\forall P_1, P_2 \in A$ *and* $b \in Base$ $[\![B]\!]^A_{bai}(P_1 \vee P_2) = [\![B]\!]^A_{bai}(P_1) \vee [\![P_2]\!]^A_{bai}$ *then:* $\forall P_2, P_2, Q \in A \quad C \in \mathbb{C}$

$$[\![C]\!]^A_{bai}(P_1) \leq Q \textit{ and } [\![C]\!]^A_{bai}(P_2) \leq Q \implies [\![C]\!]^A_{bai}(P_1 \vee P_2) \leq Q$$

Proof 3 *Let* $S_i = [\![C]\!]^A_{bai}(P_i)$ *by Theorem 4.2* $[\![C]\!]^A_{bai}(P_1 \vee P_2) = [\![C]\!]^A_{bai}(P_1) \vee [\![C]\!]^A_{bai}(P_2) \leq Q \vee Q = Q$

But requiring all the base commands to be additive is a strong requirement with $\mathcal{P}(\mathbb{S})$ the requirement is equivalent as requiring γ especially when the base commands are defined trough a Galois connection to be additive, a requiring satisfied by a very small portion of abstract domains utilized in practice.

Theorem 8 (Base commands additivity) *Given a Galois connection* $\langle D, \leq_D \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$ *and* $[\![b]\!]^D_{bai}$ *additive.*

$$\alpha \circ [\![b]\!]^D_{bai} \circ \gamma \textit{ additive} \impliedby \gamma \textit{ additive}$$

Proof 4

$$\begin{aligned}\alpha(\llbracket b\rrbracket^A_{bai}(\gamma(P_1 \vee P_2))) &= \alpha(\llbracket b\rrbracket^A_{bai}(\gamma(P_1) \vee \gamma(P_2)))\\ &= \alpha(\llbracket b\rrbracket^A_{bai}(\gamma(P_1)) \vee \llbracket b\rrbracket^A_{bai}(\gamma(P_2)))\\ &= \alpha(\llbracket b\rrbracket^A_{bai}(\gamma(P_1))) \vee \alpha(\llbracket b\rrbracket^A_{bai}(\gamma(P_2)))\end{aligned}$$

4.2.1 Local condition

Given that additivity conditions on the domain A are too restrictive we will look at local conditions.

One could think that requiring γ to be additive on $P_1 \vee P_2$ is enough giving a rule like this:

$$\frac{\vdash \langle P_1\rangle_A\ C\ \langle Q\rangle \qquad \vdash \langle P_2\rangle_A\ C\ \langle Q\rangle \qquad \gamma(P_1 \vee P_2) = \gamma(P_1) \vee \gamma(P_2)}{\vdash \langle P_1 \vee P_2\rangle_A\ C\ \langle Q\rangle}\ (merge)$$

Example 4.3 (($merge$) rule is unsound) *Let A and $\llbracket b\rrbracket^A_{bai}$ be the same as in 4.1.*

By picking $P_1 = x \in [0,1]$ and $P_2 = x \in [2,2]$ we can obtain the following judgments:

- $\langle P_1\rangle_A\ C\ \langle\bot\rangle$
- $\langle P_2\rangle_A\ C\ \langle\bot\rangle$

And by applying the ($merge$) rule we can obtain: $\langle P_1 \vee P_2\rangle_A\ C\ \langle Q\rangle$ but by running command C on $P_1 \vee P_2$ we obtain $x \in [1,1]$.

$$\begin{aligned}\llbracket C\rrbracket^A_{bai}(P_1 \vee P_2) &= \llbracket C\rrbracket^A_{bai}(x \in [0,2])\\ &= \llbracket x = 1?\rrbracket^A_{bai}(\llbracket x = 0? + x = 2?\rrbracket^A_{bai}(x \in [0,2]))\\ &= \llbracket x = 1?\rrbracket^A_{bai}(\llbracket x = 0?\rrbracket^A_{bai}(x \in [0,2]) \vee \llbracket x = 2?\rrbracket^A_{bai}(x \in [0,2]))\\ &= \llbracket x = 1?\rrbracket^A_{bai}(x \in [0,0]) \vee (x \in [2,2]))\\ &= \llbracket x = 1?\rrbracket^A_{bai}(x \in [0,2])\\ &= x \in [1,1]\end{aligned}$$

The issue is caused by the imprecision added by the $\vee$ introduced by the non deterministic choice.

In fact the following equivalence between programs that is true on the concrete domain $(C_1 + C_2) \fatsemi C_3 \equiv (C_1 \fatsemi C_3) + (C_2 \fatsemi C_3)$ it's not true in the abstract, the program $C' = (x = 0? \fatsemi x = 1?) + (x = 2? \fatsemi x = 1?)$ that should be equivalent to C it's not: $\llbracket C'\rrbracket^A_{bai}(P_1 \vee P_2) = \bot$.

The fact that programs C and C' are different when interpreted by $\llbracket\cdot\rrbracket^A_{bai}$ means that we can't look at them as a KAT, in fact this would violate the axiom

$(p+q)r = pr+qr$. It should also be noted that the dual axiom $r(p+q) = rp+rq$ instead holds.

Theorem 9 $\forall C_1 C_2 C_3$

$$[\![C_1 \,{}_{9}^{\circ}\, (C_2 + C_3)]\!]^A_{bai} = [\![(C_1 \,{}_{9}^{\circ}\, C_2) + (C_1 \,{}_{9}^{\circ}\, C_3)]\!]^A_{bai}$$

Proof 5

$$\begin{aligned}[\![C_1 \,{}_{9}^{\circ}\, (C_2 + C_3)]\!]^A_{bai}(P) &= [\![C_2]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P)) \vee [\![C_3]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P)) \\ &= [\![C_1 \,{}_{9}^{\circ}\, C_2]\!]^A_{bai}(P) \vee [\![C_1 \,{}_{9}^{\circ}\, C_3]\!]^A_{bai}(P) \\ &= [\![C_1 \,{}_{9}^{\circ}\, C_2 + C_1 \,{}_{9}^{\circ}\, C_3]\!]^A_{bai}(P)\end{aligned}$$

Requiring $(C_1 + C_2) \,{}_{9}^{\circ}\, C_3 \equiv (C_1 \,{}_{9}^{\circ}\, C_3) + (C_2 \,{}_{9}^{\circ}\, C_3)$ is equivalent to requiring the abstract semantics of every program to be additive:

Theorem 10

$$\forall C_1 C_2 C_3 \quad [\![(C_1 + C_2) \,{}_{9}^{\circ}\, C_3]\!]^A_{bai} = [\![(C_1 \,{}_{9}^{\circ}\, C_3) + (C_2 \,{}_{9}^{\circ}\, C_3)]\!]^A_{bai}$$

$$\Longleftrightarrow$$

$$\forall P_1 P_2 (decidable) C' \quad [\![C']\!]^A_{bai}(P_1 \vee P_2) = [\![C_1]\!]^A_{bai}(P_1) \vee [\![C_2]\!]^A_{bai}(P_2)$$

Proof 6

- *($\Longleftarrow$):*

$$\begin{aligned}[\![(C_1 + C_2) \,{}_{9}^{\circ}\, C_3]\!]^A_{bai}(P) &= [\![C_3]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P) \vee [\![C_2]\!]^A_{bai}(P)) \\ &= [\![C_3]\!]^A_{bai}([\![C_1]\!]^A_{bai}(P)) \vee [\![C_3]\!]^A_{bai}([\![C_2]\!]^A_{bai}(P)) \\ &= [\![(C_1 \,{}_{9}^{\circ}\, C_3) + (C_2 \,{}_{9}^{\circ}\, C_3)]\!]^A_{bai}(P)\end{aligned}$$

- *($\Longrightarrow$): For every proposition P we can construct a program $c(P)$ such that $[\![c(P)]\!]^A_{bai}(Q) = P$ (we just need to provide a program for every join-irriducible element in A)*

$$\begin{aligned}[\![C]\!]^A_{bai}(P_1 \vee P_2) &= [\![C]\!]^A_{bai}([\![c(P_1)]\!]^A_{bai}(Q) \vee [\![c(P_2)]\!]^A_{bai}(Q)) \\ &= [\![C]\!]^A_{bai}([\![c(P_1) + c(P_2)]\!]^A_{bai}(Q)) \\ &= [\![(c(P_1) + c(P_2)) \,{}_{9}^{\circ}\, C]\!]^A_{bai}(Q)) \\ &= [\![(c(P_1) \,{}_{9}^{\circ}\, C) + (c(P_2) \,{}_{9}^{\circ}\, C)]\!]^A_{bai}(Q)) \\ &= [\![C]\!]^A_{bai}([\![c(P_1)]\!]^A_{bai}(Q)) \vee [\![C]\!]^A_{bai}([\![c(P_2)]\!]^A_{bai}(Q)) \\ &= [\![C]\!]^A_{bai}(P_1) \vee [\![C]\!]^A_{bai}(P_2)\end{aligned}$$