Written exam
MPRI 2-6, year 2017–2018

29 November 2017

Duration: 3 hours (8:45–11:45)

The only documents allowed are your own printed copy of the course slides and your personal notes.
The use of connected electronic devices (computers, phones, etc.) is prohibited.
The questions are written in English. You can answer either in English or in French.
The different exercises are independent and can be solved in any order.
You should answer the exercises on different sheets of paper.
It will not be answered to any question during the exam. In case of an ambiguity or an error in the definitions or the questions, it is part of the exam to correct them and answer to the best of your abilities.

Exercise 1: Cardinal power and string abstraction

The purpose of this exercise is to study the notion of cardinal power of abstract domains (that was defined and used in the course). We will also consider the applications of this construction to the abstraction of sets of strings.

In the following, we consider a concrete domain $\mathbb{C}$ that is defined as a powerset domain, namely $\mathbb{C} = \mathcal{P}(\mathbb{E})$ for some given set $\mathbb{E}$, and is ordered by the inclusion ordering $\subseteq$. We consider two abstract domains $(\mathbb{D}_0, \sqsubseteq_0)$ and $(\mathbb{D}_1, \sqsubseteq_1)$ with monotone concretization functions $\gamma_0 : \mathbb{D}_0 \longrightarrow \mathbb{C}$ and $\gamma_1 : \mathbb{D}_1 \longrightarrow \mathbb{C}$.

We define the *cardinal power* domain of $\mathbb{D}_0$ and $\mathbb{D}_1$ as follows:

- the abstract elements are monotone functions from $\mathbb{D}_0$ to $\mathbb{D}_1$:

$$\mathbb{D}_{\rightarrow} = \mathbb{D}_0 \longrightarrow_m \mathbb{D}_1$$

- the abstract order $\sqsubseteq_{\rightarrow}$ is the pointwise extension of $\sqsubseteq_1$:

$$\phi \sqsubseteq_{\rightarrow} \phi' \iff \forall x_0 \in \mathbb{D}_0,\ \phi(x_0) \sqsubseteq_1 \phi'(x_0)$$

- the concretization function $\gamma_{\rightarrow} : \mathbb{D}_{\rightarrow} \longrightarrow \mathbb{C}$ is defined by:

$$\gamma_{\rightarrow}(\phi) = \{x \in \mathbb{E} \mid \forall x_0 \in \mathbb{D}_0,\ x \in \gamma_0(x_0) \Longrightarrow x \in \gamma_1(\phi(x_0))\}$$

Question 1 (Getting started: simple operations and abstract elements).

1. *We assume that $\mathbb{D}_1$ has an infimum $\bot_1$. Prove that $\mathbb{D}_{\rightarrow}$ has an infimum $\bot_{\rightarrow} = \lambda(x_0 \in \mathbb{D}_0) \cdot \bot_1$.*
2. *We assume that $\mathbb{D}_1$ has a binary least upper bound operation $\sqcup_1$. Prove that the pointwise extension $\sqcup_{\rightarrow}$ of $\sqcup_1$ computes the least upper bound of pairs of elements of $\mathbb{D}_{\rightarrow}$.*
3. *Prove that $\gamma_{\rightarrow}$ is monotone.*

Solution 1 *In this section, the results all follow from the pointwise ordering:*

1. *Follows directly from the definition of the pointwise ordering.*
2. *Follows directly from the definition of the pointwise ordering.*
3. *Let $\phi, \phi' \in \mathbb{D}_{\rightarrow}$ such that $\phi \sqsubseteq_{\rightarrow} \phi'$ and $x \in \mathbb{E}$. We prove that if $x \in \gamma_{\rightarrow}(\phi)$, then $x \in \gamma_{\rightarrow}(\phi')$. We assume $x \in \gamma_{\rightarrow}(\phi)$, and let $x_0 \in \mathbb{D}_0$. If $x \in \gamma_0(x_0)$, then by definition of $\gamma_{\rightarrow}$, we have $x \in \gamma_1(\phi(x_0))$, so, by definition of the pointwise ordering, and since $\phi \sqsubseteq_{\rightarrow} \phi'$, we also have $x \in \gamma_1(\phi'(x_0))$. This effectively proves that $x \in \gamma_{\rightarrow}(\phi')$. Thus, we conclude that $\gamma_{\rightarrow}$ is monotone.*

Question 2 (Application to string abstraction).

We consider an abstraction of sets of strings:

- *we let $\mathcal{A}$ denote the alphabet $\mathcal{A} = \{a, b\}$, $\mathcal{A}^\star$ denote the set of finite words over $\mathcal{A}$, and $\mathbb{E} = \mathcal{A}^\star$; furthermore, we write $|w|$ for the length of a word in $\mathcal{A}^\star$;*
- *we let $\mathbb{D}_0 = \{\bot, \top\} \cup \{a, b, aa, ab, ba, bb\}$, and $\gamma_0 : \mathbb{D}_0 \longrightarrow \mathbb{C}$ be defined by $\gamma_0(\bot) = \emptyset$, $\gamma_0(\top) = \mathbb{E}$ and, for all $w \in \{a, b, aa, ab, ba, bb\}$, $\gamma_0(w) = \{w' \in \mathbb{E} \mid w$ is a prefix of $w'\}$;*
- *we let $\mathbb{D}_1$ be the interval domain, and $\gamma_1 : \mathbb{D}_1 \longrightarrow \mathbb{C}, \bot \longmapsto \emptyset, I \longmapsto \{w \in \mathcal{A}^\star \mid |w| \in I\}$.*

1. *Explicit the abstract order over $\mathbb{D}_0$ and $\mathbb{D}_1$ (hint: functions γ_0 and γ_1 should be monotone).*
2. *We define the following abstract element:*

$$\begin{array}{rccl} \phi : & \bot & \longmapsto & \bot \\ & a & \longmapsto & [1,2] \\ & b & \longmapsto & [2,3] \\ & aa & \longmapsto & \bot \\ & ab & \longmapsto & [2,2] \\ & ba & \longmapsto & \bot \\ & bb & \longmapsto & [2,3] \\ & \top & \longmapsto & [1,3] \end{array}$$

   Give the concretization of ϕ.
3. *We consider the set of words defined below:*

$$X = \{a^k b^k \mid 1 \leq k \leq 10\}$$

   Give an abstraction of X that is as precise as possible (if you cannot choose between several incomparable abstractions, explain why). Informally explain what information is lost in this abstraction.
4. *Same question for the set below:*

$$Y = \{a^k \mid k \in \mathbb{N}\} \cup \{a^k b^k \mid 0 \leq k \leq 10\}$$

Solution 2 .

1. *The order over $\mathbb{D}_0$ is the opposite of the prefix ordering, since a prefix describes* more *words: $aa \sqsubseteq_0 a$, $ab \sqsubseteq_0 a$, $ba \sqsubseteq_0 b$, $bb \sqsubseteq_0 b$. Moreover, $\bot$ define the infimum and $\top$ the supremum of $\mathbb{D}_0$. The order relation over $\mathbb{D}_1$ is the usual order relation over the abstract domain of intervals.*
2. $\gamma_{\rightarrow}(\phi) = \{a, ab, bb, bba, bbb\}$
3. *The best abstraction is the following element:*

$$\begin{array}{rccl} \phi : & \bot & \longmapsto & \bot \\ & a & \longmapsto & [2,20] \\ & b & \longmapsto & \bot \\ & aa & \longmapsto & [4,20] \\ & ab & \longmapsto & [2,2] \\ & ba & \longmapsto & \bot \\ & bb & \longmapsto & \bot \\ & \top & \longmapsto & [2,20] \end{array}$$

   It forgets that all words in X are of even length, and, while it captures precise information about the begining and the length of words, it says nothing about what happens after the first two characters. As an example, it accepts a^2b^8. Still, it precisely expresses that no word in X starts with a b.
4. *The best abstraction is the following element:*

$$\begin{array}{rccl} \phi : & \bot & \longmapsto & \bot \\ & a & \longmapsto & [1,+\infty[\\ & b & \longmapsto & \bot \\ & aa & \longmapsto & [2,+\infty[\\ & ab & \longmapsto & [2,2] \\ & ba & \longmapsto & \bot \\ & bb & \longmapsto & \bot \\ & \top & \longmapsto & [0,+\infty[\end{array}$$

As in the previous question, length information is well preserved, as well as the fact that no word starts with a b, but all information is lost beyond the first two characters, regarding to the relations between occurrences of a and occurrences of b.

Question 3 (A condition for the existence of a Galois-connection).

In this question we consider the general case again (not necessarily the specific instances related to the abstraction of strings).

1. *Let us assume that γ_1 defines a Galois-connection, with an abstraction function α_1:*

$$\begin{tikzcd}(\mathbb{C}, \subseteq) \arrow[r, shift right, "\alpha_1"'] & (\mathbb{D}_1, \sqsubseteq_1) \arrow[l, shift right, "\gamma_1"']\end{tikzcd}$$

   Prove that $\gamma_{\rightarrow}$ also defines a Galois-connection, and give the corresponding abstraction function.
2. *Does this result apply to the abstractions considered in the previous question?*
3. *In the general case, is the existence of an abstraction function α_1 a necessary condition for the cardinal power domain to have an abstraction function? Either prove this assertion, or give a counter-example.*

Solution 3 .

1. *Let $X \in \mathbb{C}$ and $\phi \in \mathbb{D}_{\rightarrow}$. Then:*

$$\begin{array}{ll} X \subseteq \gamma_{\rightarrow}(\phi) & \\ \iff & \forall x \in X, \forall x_0 \in \mathbb{D}_0,\ x \in \gamma_0(x_0) \Longrightarrow x \in \gamma_1(\phi(x_0)) \\ \iff & \forall x_0 \in \mathbb{D}_0, \forall x \in X,\ x \in \gamma_0(x_0) \Longrightarrow x \in \gamma_1(\phi(x_0)) \\ \iff & \forall x_0 \in \mathbb{D}_0, X \cap \gamma_0(x_0) \subseteq \gamma_1(\phi(x_0)) \\ \iff & \forall x_0 \in \mathbb{D}_0, \alpha_1(X) \cap \gamma_0(x_0) \sqsubseteq_1 \phi(x_0) \\ \iff & \alpha_{\rightarrow}(X) \sqsubseteq_{\rightarrow} \phi \end{array}$$

   where:

$$\alpha_{\rightarrow}(X) = \lambda(x_0 \in \mathbb{D}_0) \cdot \alpha_1(X \cap \gamma_0(x_0))$$

2. *The abstraction relation defined by γ_1 admits an abstraction function so the result applies. It also confirms that each of the concrete sets considered in the previous question has a best abstraction (as any other).*
3. *The existence of an abstraction function α_1 is a sufficient condition but not a necessary one. As an example, let us take $\mathbb{E} = \mathbb{R}^n$, $\mathbb{D}_0 = \{\bot\} \cup \{\{v\} \mid v \in \mathbb{E}\}$, $\gamma(\bot) = \emptyset$ and $\gamma_0(\{v\}) = \{v\}$, and let $\mathbb{D}_1, \gamma_1$ define the domain of convex polyhedra. Then, $(\mathbb{D}_1, \gamma_1)$ has no best abstraction. However, any subset of $\mathbb{E}$ can be abstracted exactly by:*

$$\begin{array}{rccl} \alpha_{\rightarrow}(X): & \bot & \longmapsto & \bot_1 \\ & \{v\} & \longmapsto & \left\{ \begin{array}{ll} \bot_1 & \text{if } v \notin X \\ (\!|v|\!) & \text{if } v \in X \end{array} \right. \end{array}$$

   where $(\!|v|\!)$ describes the polyhedron that contains only one point, that is v.

Question 4 (Reduction and representation).

We now study the reduction of abstract elements, which propagates information in order to strengthen the relations they capture.

1. *We assume the same abstractions as in Question 2. What do you think of the concretization of the elements below:*

$$\begin{array}{rccl} \phi: & \bot & \longmapsto & \bot \\ & a & \longmapsto & [2, 10] \\ & b & \longmapsto & [1, 4] \\ & aa & \longmapsto & \bot \\ & ab & \longmapsto & \bot \\ & ba & \longmapsto & [1, 1] \\ & bb & \longmapsto & [1, 1] \\ & \top & \longmapsto & [1, 10] \end{array} \qquad \begin{array}{rccl} \phi': & \bot & \longmapsto & \bot \\ & a & \longmapsto & \bot \\ & b & \longmapsto & [1, 1] \\ & aa & \longmapsto & \bot \\ & ab & \longmapsto & \bot \\ & ba & \longmapsto & \bot \\ & bb & \longmapsto & \bot \\ & \top & \longmapsto & [1, 1] \end{array}$$

   Compute and compare these concretizations. Report your observations.

2. *In the following, we seek for a* reduction *operation that maps an element of $\mathbb{D}_\rightarrow$ into another one, that is optimal for $\sqsubseteq_\rightarrow$, but has the same concretization. In general, it is not always possible to define such a reduction operation.*
   Show that it is possible to define a reduction operation under the assumptions of Question 3, by a composition of $\alpha_\rightarrow$ and $\gamma_\rightarrow$ (write and prove this composition).
   Does that mean that the reduction is computable?
3. *We assume the same abstractions as in Question 2. Propose and justify a reduction formula that shows explicitly how to compute the reduction (hint: rely on the observations from the above example). What happens when applying this operator to the example abstract states ϕ, ϕ' defined in Question 4(1).*
4. *We assume the same abstractions as in Question 2, and seek for a compact representation of reduced elements, where we would keep only the images of the elements of $\mathbb{D}_0$ that effectively contribute to the expressiveness of the domain.*
   Derive from the result of Question 4(3) a more compact representation.

Solution 4 .

1. *First, let us consider the language described by ϕ. We see that a is not in the language, since words in the language and starting with an a should have length at least 2. Moreover, no word in the language has prefix aa or ab. We also remark that words with prefix ba or bb should have length at most 1, which is incompatible with the length of these prefixes. Thus, the only word in the language starting with b is b itself. As a consequence, $\gamma_\rightarrow(\phi) = \{b\}$.*
   Second, we consider ϕ'. This case is fairly obvious: $\gamma_\rightarrow(\phi') = \{b\}$.
   To conclude, both ϕ and ϕ' have the same concretizations. While ϕ' clearly defines this language, it is much less obvious in the case of ϕ, as a lot of information is hidden in the constraints.
2. *Let ϕ be an element of $\mathbb{D}_\rightarrow$. Defining the reduction of ϕ amounts to discovering an element ϕ' such that $\gamma_\rightarrow(\phi) = \gamma_\rightarrow(\phi')$, and that is minimal for $\sqsubseteq_\rightarrow$.*
   Thus, we assume that ϕ' is such that $\gamma_\rightarrow(\phi) = \gamma_\rightarrow(\phi')$. This means that $\gamma_\rightarrow(\phi) \subseteq \gamma_\rightarrow(\phi')$. Since the assumptions ensure the existence of a best abstraction function $\alpha_\rightarrow$, we can derive:
   $$\alpha_\rightarrow \circ \gamma_\rightarrow(\phi) \sqsubseteq_\rightarrow \phi'$$
   As a consequence, the ideal reduction (if it exists) is greater than $\alpha_\rightarrow \circ \gamma_\rightarrow(\phi)$.
   However, we have observed in the course that $\gamma_\rightarrow \circ \alpha_\rightarrow \circ \gamma_\rightarrow = \gamma_\rightarrow$. We conclude that the ideal reduction exists and is equal to $\alpha_\rightarrow \circ \gamma_\rightarrow(\phi)$. As a sidenote, we have not used the properties of the cardinal power abstraction, and this result would apply to any other Galois connection.
   Unfortunately, that does not mean that the reduction can be computed in the general case, as we cannot hope for computing $\gamma_\rightarrow$. So, we still have to search for more constructive versions.
3. *The reduction of an element ϕ is defined by:*
   $$\begin{array}{rlcl}
   \phi: & \bot & \longmapsto & \bot \\
   & a & \longmapsto & (\phi(aa) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(ab) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(a) \sqcap_1 [1,+\infty[) \\
   & b & \longmapsto & (\phi(ba) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(bb) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(b) \sqcap_1 [1,+\infty[) \\
   & aa & \longmapsto & \phi(aa) \sqcap_1 [2,+\infty[\\
   & ab & \longmapsto & \phi(ab) \sqcap_1 [2,+\infty[\\
   & ba & \longmapsto & \phi(ba) \sqcap_1 [2,+\infty[\\
   & bb & \longmapsto & \phi(bb) \sqcap_1 [2,+\infty[\\
   & \top & \longmapsto & (\phi(aa) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(ab) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(a) \sqcap_1 [1,+\infty[) \\
   & & & \sqcup_1(\phi(ba) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(bb) \sqcap_1 [2,+\infty[) \sqcup_1 (\phi(b) \sqcap_1 [1,+\infty[) \\
   & & & \sqcup_1(\phi(\top) \sqcap_1 [0,0])
   \end{array}$$
   To justify this formula, we simply need to compute the intersection of the concretization of ϕ with $\gamma_0(x_0)$ for each element x_0, and to apply the abstraction function to the result.
   The application of this operator to ϕ and ϕ' returns ϕ'.
4. *We can see in the above reduction that $\bot$ is always mapped to $\bot$ and thus does not need to be represented. Only $\top$ may account for the empty word ϵ, thus it can generally not be omitted. When its image does not contain 0, it would be safe to omit it as its range can be computed from the images of the other elements, thus it does not need to be represented. The other six elements must be preserved.*

Question 5 (A few transfer functions).

Finally, we consider a few abstract operations in the context of the instance of cardinal power defined in Question 2. For each operation, provide a definition that is as precise as possible (if you think there is no optimal solution, explain why). In particular, determine whether applying the operation on an abstract element that was obtained by reduction (Question 4) allows for a simpler definition.

1. *We would like to determine if the empty word may be in the language or not. Define a function* $\textbf{\textit{empty}}^\sharp : \mathbb{D}_\rightarrow \longrightarrow \{\textbf{\textit{false}}, \textbf{\textit{true}}\}$*, such that when* $\textbf{\textit{empty}}^\sharp(\phi)$ *returns* ***false****, then the empty word* ϵ *does not belong to* $\gamma_\rightarrow(\phi)$*.*
2. *Define a function* $\textbf{\textit{length}}^\sharp : \mathbb{D}_\rightarrow \longrightarrow \mathbb{D}_1$ *(we recall* $\mathbb{D}_1$ *is the domain of intervals here), and that returns an interval that over-approximates the length of all the words in the language.*
3. *Define a function* $\textbf{\textit{slice}}^\sharp : \mathbb{D}_\rightarrow \times \mathbb{N} \longrightarrow \mathbb{D}_\rightarrow$ *such that* $\gamma_\rightarrow(\textbf{\textit{slice}}^\sharp(\phi, k))$ *contains all the words of length at most* k *and that belong to* $\gamma_\rightarrow(\phi)$*.*

Solution 5 .

1. *To check if the empty word may belong to* $\gamma_\rightarrow(\phi)$*, we simply need to check if* 0 *belongs to the interval* $\phi(\top)$*, since* ϵ *is described only by* $\top$ *in* $\mathbb{D}_1$*. Reduction has no effect on the computation of this operation. To summarize,* $\textbf{\textit{empty}}^\sharp$ *does not require reduction of its argument and:*

$$\textbf{\textit{empty}}^\sharp(\phi) = \begin{cases} \textbf{\textit{true}} & \textit{if } 0 \in \phi(\top) \\ \textbf{\textit{false}} & \textit{otherwise} \end{cases}$$

2. *Let* ϕ *be an abstract element. Then, by definition the interval* $\phi(\top)$ *over-approximates the set* $\{|w| \mid w \in \gamma_\rightarrow(\phi)\}$*. As we have seen in Question 4, this over-approximation is not necessarily optimal when* ϕ *is not reduced. Then, a more precise range may be computed by reduction, as shown in the answers to that question. Otherwise, the computation of the most precise range will basically involve the reduction of the argument.*
To summarize, $\textbf{\textit{length}}^\sharp$ *requires reduction of its argument and:*

$$\textbf{\textit{length}}^\sharp(\phi) = \phi(\top)$$

3. *Let* ϕ *be an abstract element and* k *an integer.*
In general:

$$\textbf{\textit{slice}}^\sharp(\phi) = x_0 \mapsto \phi \sqcap_1 [0, k]$$

This operator will produce more precise results when its input was reduced.
However, when k is 1, this abstract state is not necessarily reduced, even ϕ *is. Indeed, let us consider* $k = 1$ *and:*

$$\phi : \begin{array}{lll} a & \longmapsto & [2,2] \\ b & \longmapsto & \bot \\ aa & \longmapsto & \bot \\ ab & \longmapsto & [2,2] \\ ba & \longmapsto & \bot \\ bb & \longmapsto & \bot \\ \top & \longmapsto & [0,2] \end{array}$$

Then, the direct image of the above $\textbf{\textit{slice}}^\sharp$ *operation would produce the state below, even though the language contains no word of length* 1*:*

$$\begin{array}{lll} a & \longmapsto & \bot \\ b & \longmapsto & \bot \\ aa & \longmapsto & \bot \\ ab & \longmapsto & \bot \\ ba & \longmapsto & \bot \\ bb & \longmapsto & \bot \\ \top & \longmapsto & [0,1] \end{array}$$

After reduction, we obtain:

$$\begin{array}{lcl} a & \longmapsto & \bot \\ b & \longmapsto & \bot \\ aa & \longmapsto & \bot \\ ab & \longmapsto & \bot \\ ba & \longmapsto & \bot \\ bb & \longmapsto & \bot \\ \top & \longmapsto & [0,0] \end{array}$$

This issue does not occur when $k = 0$ (only the empty word may remain in the language) or when $k \geq 2$.
*To conclude, **slice**$^\sharp$ should apply reduction on its output when $k = 1$.*

Exercise 2: Stack abstractions

The purpose of the exercise is to design abstractions for unbounded stacks of integer values and apply them to the analysis of functions where they model stacks of local variables in recursive procedure calls. We consider a language of control-flow graphs where arcs are labelled with commands in *cmd*, which can be regular arithmetic commands *acmd*, as seen in the course, or stack-related commands *scmd*:

$$\begin{array}{lcll} \mathit{cmd} & ::= & \mathit{acmd} \mid \mathit{scmd} & \\ & & & \\ \mathit{acmd} & ::= & X \leftarrow \mathit{exp} & \textit{(assignment into } X \in \mathbb{V}\textit{)} \\ & \mid & \mathit{exp} \bowtie 0 & \textit{(test, } \bowtie \in \{=,<,\ldots\}\textit{)} \\ & & & \\ \mathit{scmd} & ::= & \mathbf{push}\,(S, \mathit{exp}) & \textit{(push a value on top of stack } S \in \mathbb{S}\textit{)} \\ & \mid & \mathbf{pop}\; S & \textit{(remove the top of stack } S \in \mathbb{S}\textit{)} \\ & \mid & X \leftarrow \mathbf{top}\; S & \textit{(store the top of stack } S \in \mathbb{S} \textit{ into } X \in \mathbb{V}\textit{)} \\ & \mid & \mathbf{top}\; S \leftarrow \mathit{exp} & \textit{(change the value of the top of stack } S \in \mathbb{S}\textit{)} \end{array}$$

$\mathbb{V}$ is a finite set of integer-valued variables and *exp* are the usual integer expressions referring to variables (not stacks), such as: $\mathit{exp} ::= \mathbb{Z} \mid \mathbb{V} \mid -\,\mathit{exp} \mid \mathit{exp} + \mathit{exp}$.
Additionally, $\mathbb{S}$ is a finite set of (non-empty) stacks of integer values — variables and stacks are distinct, so that $\mathbb{V} \cap \mathbb{S} = \emptyset$. A new value can be pushed on top of a stack with **push** (S, exp), increasing the stack size by one. The top element of a stack can be removed with **pop** S, discarding the value and decreasing the stack size by one. Additionally, the value at the top of the stack can be copied into a variable with $X \leftarrow$ **top** S, without changing the stack contents nor its size. It can be changed with **top** $S \leftarrow \mathit{exp}$, without changing other values nor the size.

A concrete environment associates an integer to each variable in $\mathbb{V}$, and a *non-empty* sequence of integers to each stack in $\mathbb{S}$, hence the following concrete domain $\mathcal{D}$:

$$\mathcal{D} \overset{\text{def}}{=} \mathcal{P}(\mathcal{E}) \text{ where } \mathcal{E} \overset{\text{def}}{=} (\mathbb{V} \to \mathbb{Z}) \times (\mathbb{S} \to \mathbb{Z}^+)$$

We denote as $e = \langle e_1, \ldots, e_n \rangle \in \mathbb{Z}^+$ a non-empty sequence of size $|e| = n \geq 1$, and e_i the element at position i in e. Finally, $\cdot$ is the concatenation of sequences, so that, for instance, $\langle x \rangle \cdot \sigma$ corresponds to pushing the value $x \in \mathbb{Z}$ on top of the stack $\sigma \in \mathbb{Z}^+$.

A program starts in an environment where each scalar variable $V \in \mathbb{V}$ is initialized to 0, and each stack $S \in \mathbb{S}$ has a unique element equal to 0.

Question 1 (Concrete semantics).
Give the concrete semantics $\mathsf{C}[\![\, \mathit{scmd} \,]\!] : \mathcal{D} \to \mathcal{D}$ of the stack commands. We assume that removing the top of a stack with only one element is an error that stops the program, so that stacks never get empty — such environments produce no output in $\mathsf{C}[\![\ ]\!]$.
We do not ask to recall the semantics of assignments and tests $\mathsf{C}[\![\, \mathit{acmd} \,]\!] : \mathcal{D} \to \mathcal{D}$, nor the semantics $\mathsf{E}[\![\, \mathit{exp} \,]\!] : (\mathbb{V} \to \mathbb{Z}) \to \mathcal{P}(\mathbb{Z})$ of arithmetic expressions, which you can use without defining them — they are as in the course.

Correction.

- $\mathsf{C}[\![\, \textbf{push}\ (S,e) \,]\!]\, R \overset{\text{def}}{=} \{\, (\rho, \sigma[S \mapsto \langle v\rangle \cdot \sigma(S)]) \mid (\rho,\sigma) \in R,\ v \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\}$
- $\mathsf{C}[\![\, \textbf{pop}\ S \,]\!]\, R \overset{\text{def}}{=} \{\, (\rho, \sigma[S \mapsto s]) \mid (\rho,\sigma) \in R,\ \exists v \in \mathbb{Z} : \sigma(S) = \langle v\rangle \cdot s,\ |s| \geq 1 \,\}$
- $\mathsf{C}[\![\, X \leftarrow \textbf{top}\ S \,]\!]\, R \overset{\text{def}}{=} \{\, (\rho[X \mapsto v], \sigma) \mid (\rho,\sigma) \in R,\ \exists s : \sigma(S) = \langle v\rangle \cdot s \,\}$
- $\mathsf{C}[\![\, \textbf{top}\ S \leftarrow e \,]\!]\, R \overset{\text{def}}{=} \{\, (\rho, \sigma[S \to \langle v\rangle \cdot s]) \mid (\rho,\sigma) \in R,\ v \in \mathsf{E}[\![\, e \,]\!]\, \rho,\ \exists v' \in \mathbb{Z} : \sigma(S) = \langle v'\rangle \cdot s \,\}$

□

Question 2 (Set abstraction).
We first consider an abstraction that collects the set of values appearing in each stack, forgetting the order and number of occurrences of these values. The domain thus becomes:

$$\mathcal{D}^s \overset{\text{def}}{=} \mathcal{P}(\mathcal{E}^s) \textit{ where } \mathcal{E}^s \overset{\text{def}}{=} (\mathbb{V} \to \mathbb{Z}) \times (\mathbb{S} \to \mathcal{P}(\mathbb{Z}))$$

1. *Give a Galois connection (α^s, γ^s) between $(\mathcal{D}, \subseteq)$ and $(\mathcal{D}^s, \subseteq)$ — prove that it is indeed a Galois connection.*
2. *Use the Galois connection to construct best abstractions $\mathsf{C}^s[\![\, scmd \,]\!]$ of the semantics of stack commands from Question 1 and the best abstraction $\cup^s$ of the union $\cup$.*
3. *Give an example program and its semantics in $\mathcal{D}^s$ demonstrating the concept of weak update.*

Correction.

1. - $\alpha^s(R) \overset{\text{def}}{=} \{\, (\rho, \lambda S.\{\, \sigma(S)_i \mid i \leq |\sigma(S)| \,\}) \mid (\rho,\sigma) \in R \,\}$
   - $\gamma^s(Z) \overset{\text{def}}{=} \{\, (\rho,\sigma) \mid \exists(\rho,\zeta) \in Z : \forall S \in \mathbb{S} : \forall i \leq |\sigma(S)| : \sigma(S)_i \in \zeta(S) \,\}$
   - α^s and γ^s obviously monotonic, as they are complete $\cup-$morphisms.
   - $\gamma^s(\alpha^s(R))$
     $= \{\, (\rho,\sigma) \mid \exists(\rho,\zeta) \in \alpha^s(R) : \forall S \in \mathbb{S} : \forall i \leq |\sigma(S)| : \sigma(S)_i \in \zeta(S) \,\}$
     $= \{\, (\rho,\sigma) \mid \exists(\rho,\sigma) \in R : \forall S \in \mathbb{S} : \forall i \leq |\sigma(S)| : \exists j \leq |\sigma(S)| : \sigma(S)_i = \sigma(S)_j \,\}$
     $\supseteq R$
   - $\alpha^s(\gamma^s(Z))$
     $= \{\, (\rho, \lambda S.\{\, \sigma(S)_i \mid i \leq |\sigma(S)| \,\}) \mid (\rho,\sigma) \in \gamma^s(Z) \,\}$
     $= \{\, (\rho, \lambda S.\{\, \sigma(S)_i \mid i \leq |\sigma(S)| \,\}) \mid \exists(\rho,\zeta) \in Z : \forall S \in \mathbb{S} : \forall j \leq |\sigma(S)| : \sigma(S)_j \in \zeta(S) \,\}$
     $= Z$
     We have in fact a Galois insertion.
2. - $\mathsf{C}^s[\![\, \textbf{push}\ (S,e) \,]\!]\, Z = \{\, (\rho, \zeta[S \mapsto \zeta(S) \cup \{v\}]) \mid (\rho,\zeta) \in Z,\ v \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\}$
   - $\mathsf{C}^s[\![\, \textbf{pop}\ S \,]\!]\, Z = Z$
   - $\mathsf{C}^s[\![\, X \leftarrow \textbf{top}\ S \,]\!]\, Z = \{\, (\rho[X \mapsto v], \zeta) \mid (\rho,\zeta) \in Z,\ v \in \zeta(S) \,\}$
   - $\mathsf{C}^s[\![\, \textbf{top}\ S \leftarrow e \,]\!]\, Z = \mathsf{C}^s[\![\, \textbf{push}\ (S,e) \,]\!]\, Z$
   - $\cup^s = \cup$
3. Consider **push** $(S, 1)$. After the instruction, the abstract stack is $S \mapsto \{0, 1\}$. In particular, abstract abstraction, we do not know whether the top of the stack equals 0 or 1.

□

Question 3 (Uniform abstraction).
We will use, in Questions 6–8, numeric abstract domains to abstract our environments. Unfortunately, in $\mathcal{E}^s$, stack variables appear as set-valued, and not integer-valued. As a first step, we abstract here $\mathcal{D}^s$ further into $\mathcal{D}^u$, defined as follows:

$$\mathcal{D}^u \overset{\text{def}}{=} \mathcal{P}(\mathcal{E}^u) \text{ where } \mathcal{E}^u \overset{\text{def}}{=} (\mathbb{V} \to \mathbb{Z}) \times (\mathbb{S} \to \mathbb{Z})$$

through the following abstraction $\alpha^u : \mathcal{D}^s \to \mathcal{D}^u$:

$$\alpha^u(Z) \overset{\text{def}}{=} \{ (\rho, \iota) \mid \exists (\rho, \zeta) \in Z : \forall S \in \mathbb{S} : \iota(S) \in \zeta(S) \}$$

i.e., each possible combination of values from the stack contents ζ leads to a different numeric environment $(\rho, \iota) \in \mathcal{D}^u$.

1. *Give a concretization function γ^u such that (α^u, γ^u) is a Galois connection between $(\mathcal{D}^s, \subseteq)$ and $(\mathcal{D}^u, \subseteq)$ — give the proof that it is indeed a Galois connection.*
2. *Show that α^u induces an over-approximation: first exhibit $Z \in \mathcal{D}^s$ such that $\gamma^u(\alpha^u(Z))$ is strictly greater than Z; then, discuss whether such a case can be actually encountered in the analysis of a program, and discuss the precision loss.*
3. *Use the Galois connection to construct best abstractions $\mathsf{C}^u[\![\, scmd \,]\!]$ of the semantics of stack commands from Question 2, and of $\cup^u$.*

Correction.

1. $\gamma^u(I)$
   $= \cup \{ R \mid \alpha^u(R) \subseteq I \}$
   $= \{ (\rho, \zeta) \mid \alpha^u(\{(\rho, \zeta)\}) \subseteq I \}$
   $= \{ (\rho, \zeta) \mid \forall \iota \in \mathbb{S} \to \mathbb{Z} : (\forall S \in \mathbb{S} : \iota(S) \in \zeta(S)) \implies (\rho, \iota) \in I \}$
2. Consider $Z \overset{\text{def}}{=} \{([], [S \mapsto \{0, 1\}]), ([], [S \mapsto \{0, 2\}])\}$.
   Then, $\alpha^u(Z) = \{ [S \mapsto 0], [S \mapsto 1], [S \mapsto 2] \}$.
   Hence, $\gamma^u(\alpha^u(Z)) = \{ ([], [S \mapsto P]) \mid P \subseteq \{0, 1, 2\}, P \neq \emptyset \}$.
   An example program is **if** ? **then push** $(S, 1)$ **else push** $(S, 2)$ **fi**. The set abstraction will be able to deduce that, at the end of the program, the stack S contains only 1s or only 2s, in addition to 0s — withouth stating how many. The uniform abstraction will additionnaly consider stacks that contain both 0s, 1s, and 2s.
3. – $\mathsf{C}^u[\![\, \textbf{push } (S, e) \,]\!] I = \{ (\rho, \iota[S \mapsto v]) \mid (\rho, \iota) \in I,\, v \in \mathsf{E}[\![\, e \,]\!] \rho \cup \{\iota(S)\} \}$
   – $\mathsf{C}^u[\![\, \textbf{pop } S \,]\!] I = I$
   – $\mathsf{C}^u[\![\, X \leftarrow \textbf{top } S \,]\!] I = \{ (\rho[X \mapsto v], \iota) \mid (\rho, \iota[S \mapsto v]) \in I \}$
   – $\mathsf{C}^u[\![\, \textbf{top } S \leftarrow e \,]\!] I = \mathsf{C}^u[\![\, \textbf{push } (S, e) \,]\!] I$
   – $\cup^u = \cup$

□

Question 4 (Precision limitation).
Consider the following program:

```
1 :  top S ← [1, 2];
2 :  push (S, [3, 4]);
3 :  X ← top S
4 :  pop S
5 :
```

Give the semantics, at each program point 1–5, of the program according to the set abstraction in $\mathcal{D}^s$ and to the uniform abstraction in $\mathcal{D}^u$ — recalling that programs start with stacks $\langle 0 \rangle$ contanining a single 0 element. Discuss the precision of these two semantics.
In particular, is it possible to infer that X and the top of the stack S are equal at the begining of line 4?

Correction.
In $\mathcal{D}^s$:
1 : $\{\, ([X \mapsto 0],\, [S \mapsto \{0\}]) \,\}$
2 : $\{\, ([X \mapsto 0],\, [S \mapsto \{0,1,2\}]) \,\}$
3 : $\{\, ([X \mapsto 0],\, [S \mapsto \{0,1,2,3,4\}]) \,\}$
4 : $\{\, ([X \mapsto x],\, [S \mapsto \{0,1,2,3,4\}]) \mid x \in \{0,1,2,3,4\} \,\}$
5 : $\{\, ([X \mapsto x],\, [S \mapsto \{0,1,2,3,4\}]) \mid x \in \{0,1,2,3,4\} \,\}$
And in $\mathcal{D}^u$:
1 : $\{\, ([X \mapsto 0],\, [S \mapsto 0]) \,\}$
2 : $\{\, ([X \mapsto 0],\, [S \mapsto s]) \mid s \in \{0,1,2\} \,\}$
3 : $\{\, ([X \mapsto 0],\, [S \mapsto s]) \mid s \in \{0,1,2,3,4\} \,\}$
4 : $\{\, ([X \mapsto x],\, [S \mapsto s]) \mid x, s \in \{0,1,2,3,4\} \,\}$
5 : $\{\, ([X \mapsto x],\, [S \mapsto s]) \mid x, s \in \{0,1,2,3,4\} \,\}$
At the end of program, we have $X \in [0,4]$ in both semantics, which is far less precise than the expected $X \in [3,4]$. Actually, the abstract results have the same concretization in $\mathcal{D}$.
Moreover, we don't have $X = S$. In particular, in $\mathcal{D}^u$, the value of x and s are chosen independently. In might be tempting to define, in Question 3, $\mathsf{C}^u[\![\, X \leftarrow \textbf{top}\ S \,]\!]\, I = \{\, (\rho[X \mapsto \iota(v)], \iota) \mid (\rho, \iota) \in I \,\}$, but this rule is unsound as it would create an equality between X and S, that does not always hold.
□

Question 5 (Strong updates on the top of stacks).
In order to solve the precision issues detected in Question 4, we consider an alternate abstraction of $\mathcal{D}$ that abstracts separately the top of each stack from the rest of the stack, which is abstracted uniformly. We thus replace $\mathcal{D}^u$ with $\mathcal{D}^\top$ as follows:

$$\mathcal{D}^\top \overset{\text{def}}{=} \mathcal{P}(\mathcal{E}^\top) \text{ where } \mathcal{E}^\top \overset{\text{def}}{=} (\mathbb{V} \to \mathbb{Z}) \times (\mathbb{S}_\top \to \mathbb{Z}) \times (\mathbb{S}_\sim \to \mathbb{Z})$$

where $\mathbb{S}_\top$ is a set of variables $\{\, S_\top \mid S \in \mathbb{S} \,\}$ denoting the top element of each stack $S \in \mathbb{S}$, and $\mathbb{S}_\sim$ is a set of variables $\{\, S_\sim \mid S \in \mathbb{S} \,\}$ summarizing the elements in the tail of each stack.

1. *Give a concretization function $\gamma^\top$ from $\mathcal{D}^\top$ to $\mathcal{D}$.*
2. *Discuss the existence of a best abstraction function $\alpha^\top$ — you can consider the case of a stack with only one element.*
3. *Give abstract operators $\mathsf{C}^\top[\![\, scmd \,]\!]$ for the stack commands in $\mathcal{D}^\top$.*
   Justify, in particular, that $\mathsf{C}^\top[\![\, X \leftarrow \textbf{top}\ S \,]\!]$ and $\mathsf{C}^\top[\![\, \textbf{top}\ S \leftarrow e \,]\!]$ can be reduced to simple assignments, while $\mathsf{C}^\top[\![\, \textbf{push}\ (S, e) \,]\!]$ and $\mathsf{C}^\top[\![\, \textbf{pop}\ S \,]\!]$ cannot.
   Also give an abstraction of the initial state.
4. *Give the semantics of the program from Question 4 in this new semantics, at each program point 1–5. Comment on the precision of the results.*

Correction.

1. $\gamma^\top(I) \overset{\text{def}}{=} \{\, (\rho, \sigma) \mid \forall f : (\forall S : 1 \leq f(S) \leq |\sigma(S)|) \implies \exists \sigma' : (\rho, \lambda S.\sigma(S)_1, \lambda S.\sigma'(S)_{f(S)}) \in I \,\}$
   where $\forall S : \exists v, v', s : \sigma(S) = \langle v \rangle \cdot s \wedge \sigma'(S) = \langle v' \rangle \cdot s$.
   Note that we define σ' so that $\mathbb{S}_\sim$ does not impose any constraint on the top of a stack, only $\mathbb{S}_\top$ does. This cannot be achieved through quantifying over $\forall f : (\forall S : 1 < f(S) \leq |\sigma(S)|)$ if we also want to handle the case of stacks with only one element.

2. Consider $R \overset{\text{def}}{=} \{[],\, [S \mapsto \langle 0 \rangle]\}$.
   Given our choice of $\gamma^\top$, any value for $S_\sim$ is acceptable. Hence, both $I_1 \overset{\text{def}}{=} \{([],\, [S_\top \mapsto 0],\, [S_\sim \mapsto 1])\}$ and $I_2 \overset{\text{def}}{=} \{([],\, [S_\top \mapsto 0],\, [S_\sim \mapsto 2])\}$ are sound abstractions of R, neither is better than the other, and $I_1 \cap I_2 = \emptyset$, i.e., there is no best abstraction.
   This leaves open, in particular, the abstraction of the initial state, which we define as:
   $\{(\lambda X.0, \lambda S_\top.0, \lambda S_\sim.0)\}$.

3. – $\mathsf{C}^\top[\![\,\mathbf{push}\ (S,e)\,]\!]\, I = \{\, (\rho, t[S_\top \mapsto v], \iota[S_\sim \mapsto v']) \mid (\rho,t,\iota) \in I,\ v \in \mathsf{E}[\![\, e\,]\!]\,\rho,\ v' \in \{t(S), \iota(S)\}\,\}$
   – $\mathsf{C}^\top[\![\,\mathbf{pop}\ S\,]\!]\, I = \{\, (\rho, t[S_\top \mapsto v], \iota) \mid (\rho, t, \iota[S_\sim \mapsto v]) \in I\,\}$
   – $\mathsf{C}^\top[\![\, X \leftarrow \mathbf{top}\ S\,]\!]\, I = \{\, (\rho[X \mapsto t(S)], t, \iota) \mid (\rho,t,\iota) \in I\,\} = \mathsf{C}^\top[\![\, X \leftarrow S_\top\,]\!]\, I$
   – $\mathsf{C}^\top[\![\,\mathbf{top}\ S \leftarrow e\,]\!]\, I = \{\, (\rho, t[S_\top \mapsto v], \iota) \mid (\rho,t,\iota) \in I,\ v \in \mathsf{E}[\![\, e\,]\!]\,\rho\,\} = \mathsf{C}^\top[\![\, S_\top \leftarrow e\,]\!]\, I$
   – $\cup^\top = \cup$

4. 1 : $\{\, ([X \mapsto 0],\ [S_\top \mapsto 0],\ [S_\sim \mapsto 0])\,\}$
   2 : $\{\, ([X \mapsto 0],\ [S_\top \mapsto t],\ [S_\sim \mapsto 0]) \mid t \in \{1,2\}\,\}$
   3 : $\{\, ([X \mapsto 0],\ [S_\top \mapsto t],\ [S_\sim \mapsto s]) \mid s \in \{0,1,2\},\ t \in \{3,4\}\,\}$
   4 : $\{\, ([X \mapsto t],\ [S_\top \mapsto t],\ [S_\sim \mapsto s]) \mid s \in \{0,1,2\},\ t \in \{3,4\}\,\}$
   5 : $\{\, ([X \mapsto x],\ [S_\top \mapsto t],\ [S_\sim \mapsto s]) \mid x \in \{3,4\}, t, s \in \{0,1,2\}\,\}$
   We are able to establish that $X \in [3,4]$.
   Note that **pop** is not equivalent to $S_\top \leftarrow S_\sim$ because, after the instruction, although $S_\top$ and $S_\sim$ have the same range, they are not necessarily equal.

□

Question 6 (Interval abstraction).
We consider now the abstraction $\mathcal{D}^\sharp$ of $\mathcal{D}^\top$ in the interval domain:

$$\mathcal{D}^\sharp \stackrel{\text{def}}{=} (\mathbb{V} \cup \mathbb{S}_\top \cup \mathbb{S}_\sim) \to \mathbb{I}$$

where $\mathbb{I}$ denotes the set of intervals of integers.

1. *Give the abstractions $\mathsf{C}^\sharp[\![\, scmd\,]\!]$ of stack commands in $\mathcal{D}^\sharp$.*
   You can assume that the interval abstraction of expressions $\mathsf{E}^\sharp[\![\, exp\,]\!]$ and that $\cup^\sharp$ are given.
2. *Give the semantics of the program from Question 4, at each program point 1–5, in the interval semantics. Discuss the precision of the results.*

Correction.

1. – $\mathsf{C}^\sharp[\![\,\mathbf{push}\ (S,e)\,]\!]\, I^\sharp = I^\sharp[S_\top \mapsto \mathsf{E}^\sharp[\![\, e\,]\!]\, I^\sharp,\ S_\sim \mapsto I^\sharp(S_\sim) \cup^\sharp I^\sharp(S_\top)]$
   – $\mathsf{C}^\sharp[\![\,\mathbf{pop}\ S\,]\!]\, I^\sharp = I^\sharp[S_\top \mapsto I^\sharp(S_\sim)]$
   – $\mathsf{C}^\sharp[\![\, X \leftarrow \mathbf{top}\ S\,]\!]\, I^\sharp = I^\sharp[X \mapsto I^\sharp(S_\top)]$
   – $\mathsf{C}^\sharp[\![\,\mathbf{top}\ S \leftarrow e\,]\!]\, I^\sharp = I^\sharp[S_\top \mapsto \mathsf{E}^\sharp[\![\, e\,]\!]\, I^\sharp]$

2. 1 : $([X \mapsto [0,0]],\ [S_\top \mapsto [0,0]],\ [S_\sim \mapsto [0,0]])$
   2 : $([X \mapsto [0,0]],\ [S_\top \mapsto [1,2]],\ [S_\sim \mapsto [0,0]])$
   3 : $([X \mapsto [0,0]],\ [S_\top \mapsto [3,4]],\ [S_\sim \mapsto [0,2]])$
   4 : $([X \mapsto [3,4]],\ [S_\top \mapsto [3,4]],\ [S_\sim \mapsto [0,2]])$
   5 : $([X \mapsto [3,4]],\ [S_\top \mapsto [0,2]],\ [S_\sim \mapsto [0,2]])$
   We do not lose any precision with repect to the uniform abstaction $\mathcal{D}^u$.

□

Question 7 (Polyhedral abstraction).
We consider now the abstraction $\mathcal{D}^\sharp$ of $\mathcal{D}^\top$ in the polyhedra domain. An element in $\mathcal{D}^\sharp$ encodes a set of points over the variables $\mathbb{V} \cup \mathbb{S}_\top \cup \mathbb{S}_\sim$.

1. *Give the abstractions $\mathsf{C}^\sharp[\![\, scmd\,]\!]$ of stack commands in the polyhedra domain.*
   State whether these abstractions are exact or not.
   You can assume that the polyhedral abstractions of arithmetic assignments and tests $\mathsf{C}^\sharp[\![\, acmd\,]\!]$ and that $\cup^\sharp$ are given.

2. *Consider the following program:*

$$\begin{array}{rl} 1: & I \leftarrow [0,+\infty]; \\ 2: & \textbf{while } X < I \textbf{ do } X \leftarrow X+1;\ \textbf{push } (S,X) \textbf{ od}; \\ 3: & \textbf{pop } S; \\ 4: & Y \leftarrow \textbf{ top } S \\ 5: & \end{array}$$

Give the analysis of this program in the polyhedra domain (using the standard polyhedra widening, delayed by one iteration).
Can you infer a relationship between X and Y? Between X and the contents of the stack S?

Correction.

1. – $\mathsf{C}^\sharp[\![\, \textbf{push } (S,e) \,]\!]\, P^\sharp = \mathsf{C}^\sharp[\![\, S_\top \leftarrow e \,]\!]\, (P^\sharp \cup^\sharp \mathsf{C}^\sharp[\![\, S_\sim \leftarrow S_\top \,]\!]\, P^\sharp)$.
   – $\mathsf{C}^\sharp[\![\, \textbf{pop } S \,]\!]\, P^\sharp$ first removes $S_\top$ from $P^\sharp$ by projection and, for each constraints containing $S_\sim$, adds a new constraint where $S_\sim$ is replaced with $S_\top$. The abstraction is exact.
   – $\mathsf{C}^\sharp[\![\, X \leftarrow \textbf{top } S \,]\!] = \mathsf{C}^\sharp[\![\, X \leftarrow S_\top \,]\!]$. Exact.
   – $\mathsf{C}^\sharp[\![\, \textbf{top } S \leftarrow e \,]\!] = \mathsf{C}^\sharp[\![\, S_\top \leftarrow e \,]\!]$. Exact.

2. At 1, we have: $X = I = S_\top = S_\sim = 0$.
   At 2, we have: $X = S_\top = S_\sim = 0$, $I \geq 0$.
   After execution of the loop body, we have: $I > 0$, $X = S_\top = 1$, $S_\sim = 0$.
   After the join, $I \geq 0$, $S_\top = X \in [0,1]$, $X \leq I$, $S_\sim = 0$.
   After a second execution of the loop body, we have: $I > 0$, $X \in [1,2]$, $X \leq I$, $S_\top = X$, $0 \leq S_\sim < X$.
   The second join gives: $I \geq 0$, $X \in [0,2]$, $X \leq I$, $S_\top = X$, $S_\sim \in [0,1]$, $S_\sim < X$.
   Applying the widening gives: $I \geq 0$, $0 \leq X \leq I$, $S_\top = X$, $0 \leq S_\sim < X$, which is stable.
   At 3, we get: $I \geq 0$, $S_\top = X = I$, $0 \leq S_\sim < I$.
   At 4, we get: $I \geq 0$, $X = I$, $0 \leq S_\sim < I$, $0 \leq S_\top < I$.
   At 5, we get: $I \geq 0$, $X = I$, $0 \leq S_\sim < I$, $0 \leq Y = S_\top < I$.
   Hence, $X > Y$, and X is greater than all stack elements ($X > S_\sim$, $X > S_\top$).

□

Question 8 (Application to inter-procedural analysis).
Consider a program composed of a set P of possibly recursive procedures.
To simplify, each procedure $p \in P$ has a single formal argument $X_p \in \mathbb{V}$, and there are no local variables; all the other variables are global. We enrich the arithmetic commands acmd with a call instruction:

$$\textit{ccmd} ::= \textbf{call } (p, \textit{exp})$$

where $p \in P$ is a procedure and exp is the actual argument. Each procedure $p \in P$ is defined by a control-flow graph $G_p = (L_p, A_p, e_p, x_p)$, where L_p is a set of nodes, and $A_p \subseteq L_p \times (\textit{acmd} \cup \textit{ccmd}) \times L_p$ is a set of arcs labelled by assignments, tests, and calls, and $e_p \in L_p$ and $x_p \in L_p$ are respectively the entry and exit point of the procedure.

1. *Show how to transform the set of graphs G_p into a single control-flow graph $G = (L, A, e, x)$ over instructions in cmd, without any call.*
   You will model each formal argument X_p using a stack S_p, which is pushed at each call and poped at each return.

2. *Consider the following recursive procedure f:*

$$f(X): \textbf{ if } X > 0 \textbf{ then } f(X-1) \textbf{ else } R \leftarrow X \textbf{ fi}$$

where the global variable R is used to model a return value.
Give the transformation into a control-flow graph without calls, according to your answer to the previous question.
Give the analysis in the polyhedra domain of the following program:

$$I \leftarrow [-100, 100];\ f(I)$$

Correction.

1. $L \overset{\text{def}}{=} \cup_{p \in P} L_p$
   Keep each arc $(\ell, c, \ell') \in A_p$ when $c \in acmd$.
   Replace each arc $(\ell, \textbf{call}\ (p, e), \ell') \in A_q$ with:

   – $(\ell, \textbf{push}\ (S_p, e), \ell'')$
   – $(\ell'', X_p \leftarrow \textbf{top}\ S, e_p)$
   – $(x_p, \textbf{pop}\ S_p, \ell')$.

   where ℓ'' is a fresh label.

□