

General Information

Detailed information about the lecture, tutorials and homework assignments can be found on the lecture website¹. Solutions have to be submitted to Moodle². Make sure your uploaded documents are readable. Blurred images will be rejected. Use Piazza³ to ask questions and discuss with your fellow students.

Assignment 1.1 (L) Recap: Implications

Remember that an implication $A \implies B$ states an *if-then* relation: If A is satisfied, then B is satisfied as well. Let $x, y \in \mathbb{Z}$ and let A, B be arbitrary formula. Decide, which of the following implications hold:

1. ☒ $x = 1 \implies 0 < x$

2. ☐ $x < 6 \implies x = 3$

3. ☒ $x > 0 \implies x \geq 0$

4. ☒ $x = -2 \implies x < -1 \vee x > 1$

5. ☒ $x = 0 \vee x = 7 \implies 4 \neq x$

6. ☐ $x = 1 \implies x \leq 3 \wedge y > 0$

7. ☒ $x < 8 \wedge y = x \implies y \neq 12$

8. ☐ $x = 1 \vee y = 1 \implies x > 0$

9. ☐ $x \neq 5 \implies false$

10. ☐ $true \implies x \neq y$

11. ☒ $false \implies x = 1$

12. ☐ $x \geq 1 \implies 2x + 3 = 5$

13. ☒ $A \wedge x = y \implies A$

14. ☒ $B \implies A \vee B$

15. ☒ $A \implies (B \implies A)$

16. ☐ $(A \implies B) \implies A$

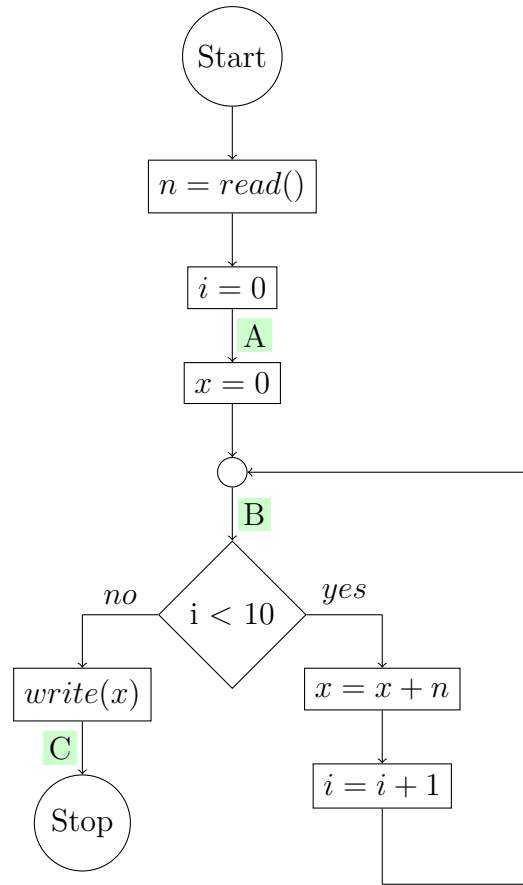
¹<https://www.in.tum.de/i02/lehre/wintersemester-1819/vorlesungen/functional-programming-and-verification/>

²<https://www.moodle.tum.de/course/view.php?id=44932>

³<https://piazza.com/tum.de/fall2018/in0003/home>

Assignment 1.2 (L) Assertions

Consider the following control flow graph:



1. Which of the following assertions hold at point *A*?

- | | | |
|--|--|---|
| (a) <input checked="" type="checkbox"/> $i \geq 0$ | (c) <input type="checkbox"/> $i \leq 10 \wedge x \neq 0$ | (e) <input checked="" type="checkbox"/> $i = 0$ |
| (b) <input type="checkbox"/> $x = 0$ | (d) <input checked="" type="checkbox"/> $true$ | (f) <input type="checkbox"/> $x = i$ |

2. Which of the following assertions hold at point *B*?

- | | | |
|---|--|--|
| (a) <input type="checkbox"/> $x = 0 \wedge i = 0$ | (c) <input type="checkbox"/> $i < x$ | (e) <input type="checkbox"/> $i \geq 0 \wedge x \geq 0$ |
| (b) <input type="checkbox"/> $x = i$ | (d) <input checked="" type="checkbox"/> $0 \leq i \leq 10$ | (f) <input checked="" type="checkbox"/> $n = 1 \implies x = i$ |

3. Which of the following assertions hold at point *C*?

- | | | |
|--|---|---|
| (a) <input checked="" type="checkbox"/> $i \geq 0$ | (c) <input checked="" type="checkbox"/> $i > 0$ | (e) <input checked="" type="checkbox"/> $x = 10n$ |
| (b) <input checked="" type="checkbox"/> $i = 10$ | (d) <input type="checkbox"/> $x \neq n$ | (f) <input checked="" type="checkbox"/> $x = i * n \wedge i = 10$ |

Assignment 1.3 (L) The Strong and the Weak

Again consider the assertions that hold at point C of assignment 1.2. Discuss the following questions:

1. When annotating the control flow graph, can you say that one of the given assertions is “better” than the others?
2. Can you arrange the given assertions in a meaningful order?
3. How can you define a *stronger than* relation formally?
4. How do *true* and *false* fit in and what is their meaning as an assertion?
5. What are the strongest assertions that still hold at A, B and C ?

Suggested Solution 1.3

1. When we want to make a statement about the state (the values of variables) at a program point, the “best” assertion is the one that gives the most precise description of the values that are actually possible in the program. Therefore, an assertion $i = 10$ is “better” than $i \geq 0$, because the information it provides is more precise. We thus say that $i = 10$ is *stronger* than $i \geq 0$. Intuitively, $x = i * n \wedge i = 10$ is the strongest among the given assertions.

2. Following the above argument, we can at least say the following:

- $x = i * n \wedge i = 10$ stronger than $i = 10$ stronger than $i > 0$ stronger than $i \geq 0$
- $x = i * n \wedge i = 10$ stronger than $x = 10n$

However, one cannot say that $i = 10$ is *stronger* than $x = 10n$ or vice versa, since the information they provide is not “better” or “worse”, but just different or incomparable.

3. In some sense, an assertion A is *stronger* than an assertion B , if A extends or includes the information of B . In other words, whenever A is satisfied, B is satisfied as well. Hence,

$$A \Rightarrow A \text{ itself} \quad A \text{ stronger than } B := A \Rightarrow B$$

is a partial order (reflexive, transitive and anti-symmetric). We can now rewrite the above statements:

- $(x = i * n \wedge i = 10 \Rightarrow i = 10) \wedge (i = 10 \Rightarrow i > 0) \wedge (i > 0 \Rightarrow i \geq 0)$
- $x = i * n \wedge i = 10 \Rightarrow x = 10n$
- Neither $i = 10 \Rightarrow x = 10n$, nor $i = 10 \Leftarrow x = 10n$

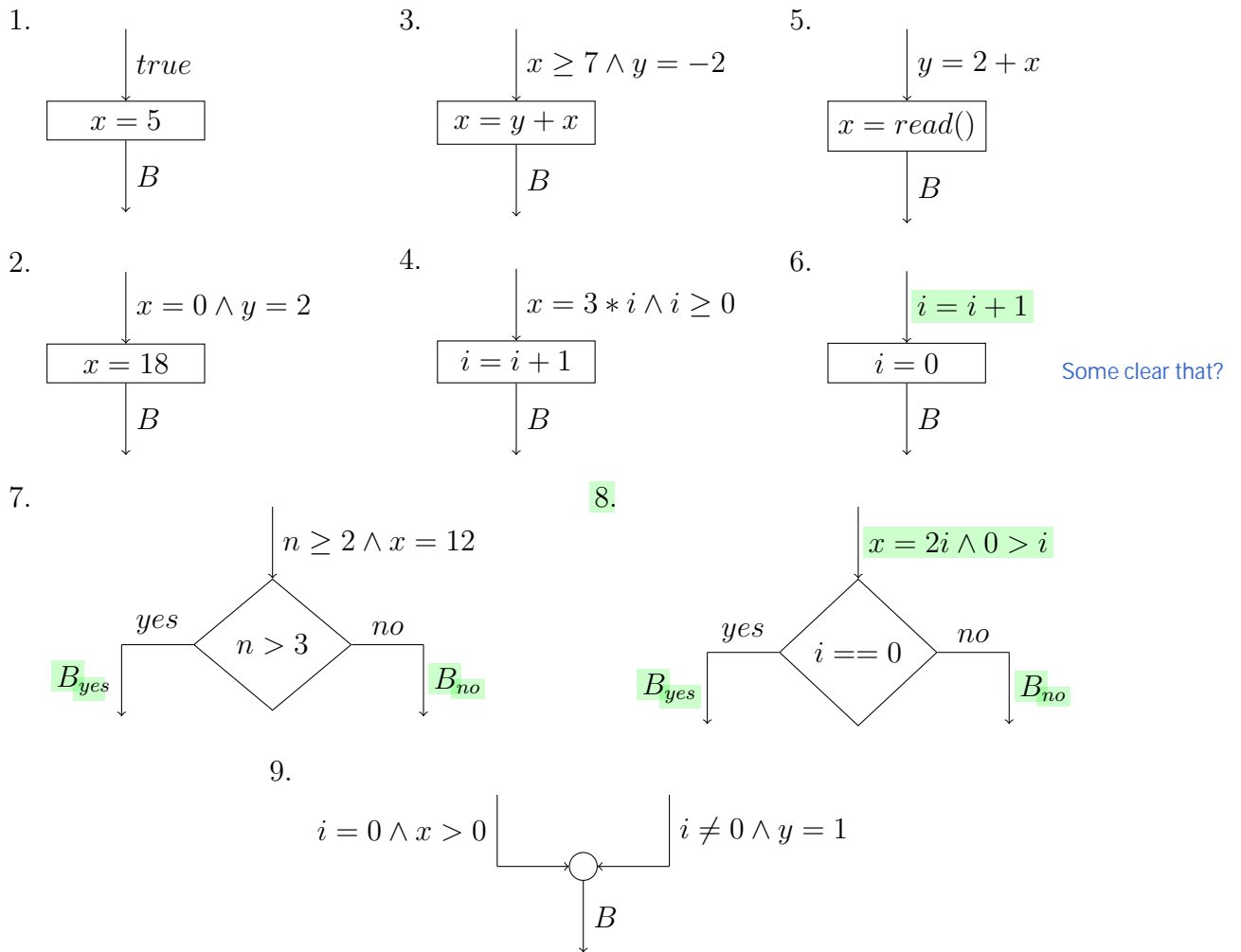
4. Since $false \Rightarrow A$ and $A \Rightarrow true$ hold for all A , *false* and *true* are the strongest and weakest assertions, respectively. While *true* is always satisfied, there is no assignment of variable values that can ever satisfy *false*. The only way for a program not to violate an assertion *false* is to never reach the respective program point.

5. For A and C the strongest assertions are already given with $i = 0$ and $x = i * n \wedge i = 10$, respectively. However, for point B we can still find a stronger assertion:
 $x = i * n \wedge 0 \leq i \leq 10$

Assignment 1.4 (L) Strongest Postconditions

Using our understanding of strong and weak assertions, it is now possible to look at statements individually and, given an assertion A that holds before a statement s , find the strongest assertion B that holds after the statement. We call B the *strongest postcondition* (of A at statement s).

For each of the following statements, find the *strongest postcondition* B :



Suggested Solution 1.4

1. $B \equiv x = 5$
2. $B \equiv x = 18 \wedge y = 2$
3. $B \equiv x \geq 5 \wedge y = -2$
4. $B \equiv x = 3(i - 1) \wedge i \geq 1$
5. $B \equiv true$

6. $B \equiv false$
7. $B_{yes} \equiv n > 3 \wedge x = 12$ and $B_{no} \equiv (n = 2 \vee n = 3) \wedge x = 12$
8. $B_{yes} \equiv false$ and $B_{no} \equiv x = 2i \wedge 0 > i$
9. $B \equiv (i = 0 \wedge x > 0) \vee (i \neq 0 \wedge y = 1)$
or $B \equiv (i = 0 \implies x > 0) \wedge (i \neq 0 \implies y = 1)$

Assignment 1.5 (H) Proving Ground: Implications

[4 Points]

Prove or refute the following claims. Whenever you rewrite any terms, annotate the rule that was used.

1. It exists an A such that

- $x < 0 \wedge y = 2 \implies A$ and
- $A \implies x < -2 \wedge y > 0$

are both satisfied.

2. No A and B exist such that

- $A \implies B \wedge x > 2$,
- $B \implies x > 5$ and
- $x > 2 \implies A$

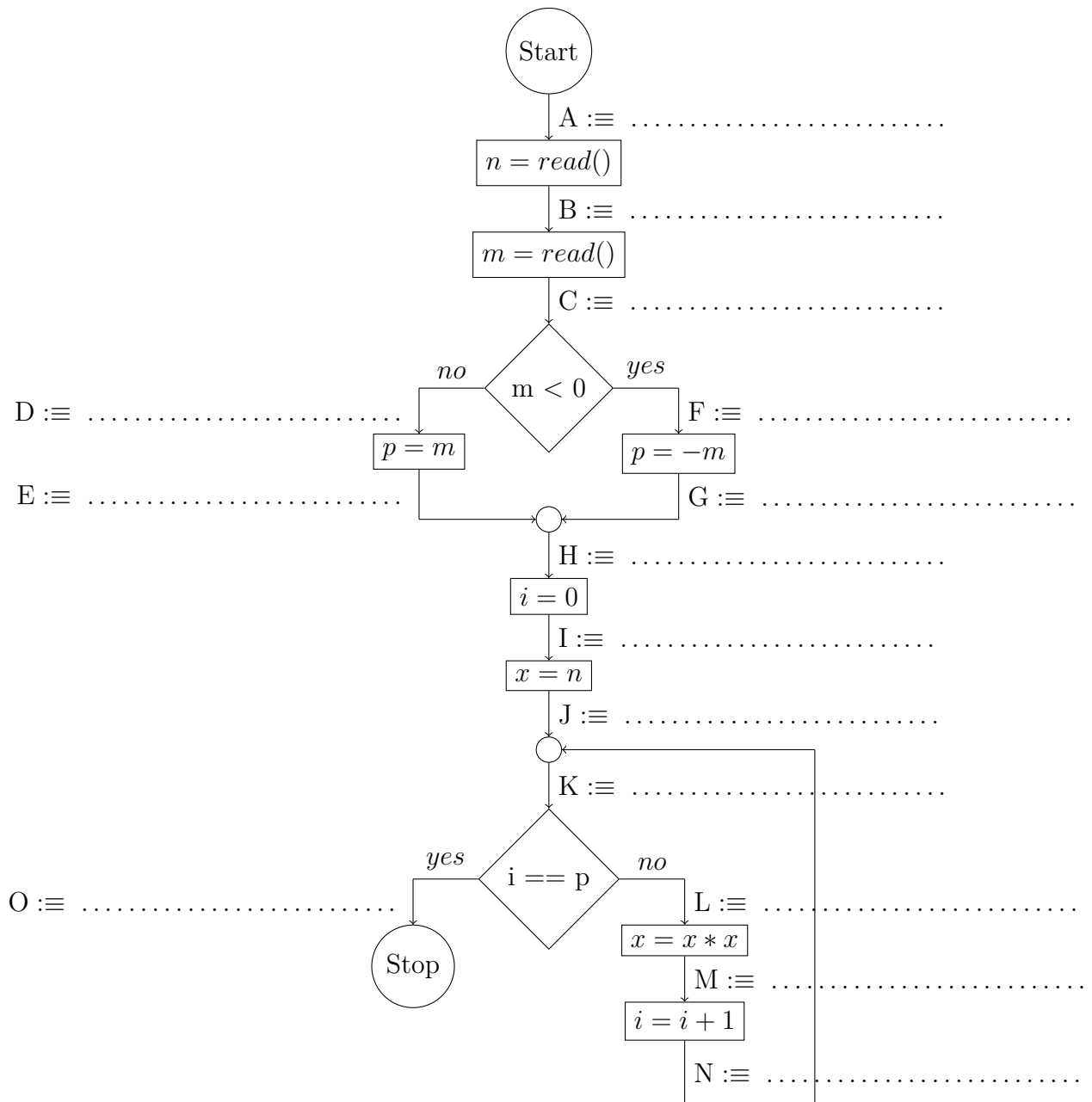
hold.

3. $(A \wedge B \implies C) \iff B \vee C \equiv A \wedge \neg C \implies \neg B$
4. $(x < 0 \implies A) \wedge (\neg A \implies x < -2) \implies (y = 3 \implies A) \equiv true$

Assignment 1.6 (H) Powerful Assertions

[7.5 Points]

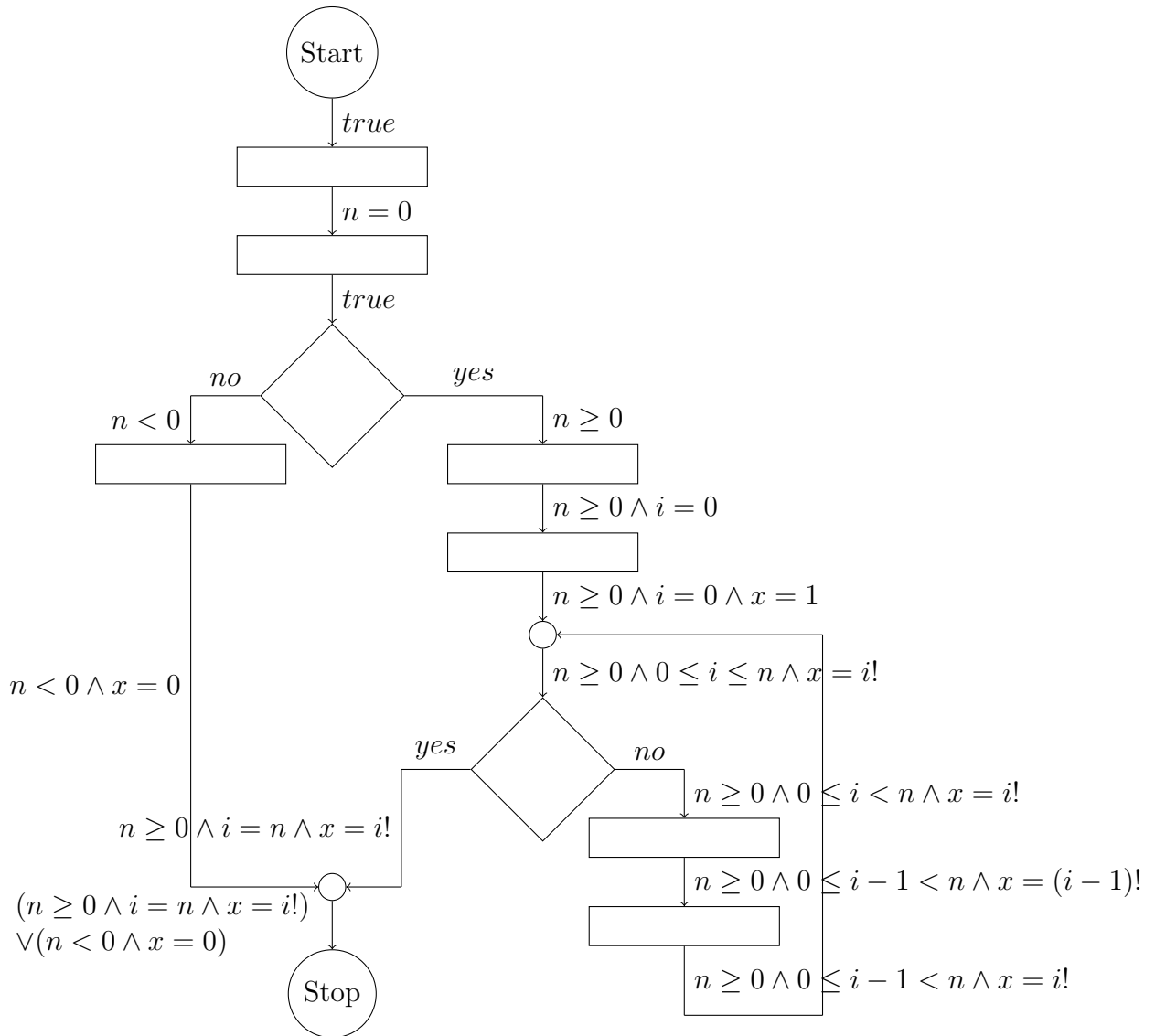
In the following control flow graph, find the strongest assertions $A - O$ that hold at the correspondingly labeled edge:



Assignment 1.7 (H) Lost Statements

[4.5 Points]

The following control flow graph's edges are annotated with the strongest assertions that hold at the respective program point. Find the statements and conditions of the program.



Assignment 1.8 (H) Return of the Penguins

[4 Points]

One day, while wandering around in the furthest corners of the university to find your exercise room, you walk by some laboratory, where a small penguin girl is tinkering on a strange device. After taking note of you watching her, she excitedly starts talking to you.

Hi there! I am Julia. I'm an informatics student! We penguins loooooooooove informatics! But now that I'm in third semester, I miss all my penguin friends from last year *sad*. There were so many other penguins, but now they are all gone *sad*. And that is why I built thiiiiiiis time machine *excitedly pointing to the device*. Everything is ready for the first test, but before I can go back to the wonderful penguin times, I need to compute the correct parameters for the flux capacitor. I was planning to write a program for that, but sadly, I only learned how to write a compiler so far *sad*.



What did you say? You can help me with the programming and you already know about assertions?! Wow, this is aweeeesome! Then I can give you the formal specification of my program and you can write it for me: The program reads two inputs a and b , that must not be modified within the program, once they are read. The program then computes the flux capacitor parameter p such that

$$(a > b \implies p = -1) \wedge (a \leq b \implies (a \geq 0 \implies p = \frac{b!}{a!} * \sum_{k=a}^b k) \wedge (a < 0 \implies p = \sum_{k=0}^b k))$$

holds at the end of the program. Can you help me?

Help Julia and implement this program in MiniJava. Inputs a and b must not be modified within the program once they are read from the user.