First tutorial

Thursday, **3.11.2016**, 13:30, MPI-SWS building (26), room 113

The permanent day/time is still to be decided.
Recap from lecture 1

A complete lattice is a partial order \((D, \leq)\) in which for every subset \(X \subseteq D\) there exist

\[
\begin{align*}
\bigvee X & \quad (\text{least upper bound for } X, \text{ join of } X) \\
& \quad \text{for all } x \in X \ (\bigvee X \text{ is an upper bound for } X) \\
& \quad \text{for all upper bounds } u \text{ for } X
\end{align*}
\]

\[
\begin{align*}
\bigwedge X & \quad (\text{greatest lower bound for } X, \text{ meet of } X) \\
& \quad \text{for all } x \in X \ (\bigwedge X \text{ is a lower bound for } X) \\
& \quad \text{for all lower bounds } l \text{ for } X
\end{align*}
\]
Recap from lecture 1

\[ d \in D \text{ is a } \textbf{fixpoint} \text{ of a function } f : D \to D \text{ if } f(d) = d \]

\textbf{Theorem (Knaster, Tarski’55)}
Let \((D, \leq)\) be a complete lattice and let \(f : D \to D\) a monotonic function. Then,

(1) there exists a (unique) \textbf{least fixpoint} of \(f\) given by
\[
\text{lfp}(f) := \bigcap \{ x \in D \mid f(x) \leq x \}
\]

(2) there exists a (unique) \textbf{greatest fixpoint} of \(f\) given by
\[
\text{gfp}(f) := \bigcup \{ x \in D \mid x \leq f(x) \}
\]
Recap from lecture 1

Let $(D, \leq)$ be a complete lattice.

A function $f : D \rightarrow D$ is $\sqcup$-continuous if for every chain $C$ in $D$,

$$f(\bigsqcup C) = \bigsqcup \{ f(c) \mid c \in C \}.$$ 

A function $f : D \rightarrow D$ is $\sqcap$-continuous if for every chain $C$ in $D$,

$$f(\bigmeet C) = \bigmeet \{ f(c) \mid c \in C \}.$$
Recap from lecture 1

Let \((D, \leq)\) be a complete lattice.

\((D, \leq)\) satisfies the \textbf{ascending chain condition (ACC)} if for every \(c_0 \leq c_1 \leq \ldots\) there exists \(n \in \mathbb{N}\) such that \(c_i = c_n\) for \(i \geq n\).

\((D, \leq)\) satisfies the \textbf{descending chain condition (DCC)} if for every \(c_0 \geq c_1 \geq \ldots\) there exists \(n \in \mathbb{N}\) such that \(c_i = c_n\) for \(i \geq n\).

\textbf{Theorem}

Let \((D, \leq)\) be a \textit{complete lattice} and let \(f : D \to D\) be a \textit{monotonic} function. Then

(1) if \((D, \leq)\) satisfies (ACC), then \(f\) is \(\sqcup\)-continuous.

(2) if \((D, \leq)\) satisfies (DCC), then \(f\) is \(\sqcap\)-continuous.
Theorem (Knaster, Tarski and Kleene)

Let \((D, \leq)\) be a complete lattice and let \(f : D \rightarrow D\) be a monotonic function.

(1) if \(f\) is \(\sqcup\)-continuous, then

\[
\text{lfp}(f) = \bigcup \{ f^i(\bot) \mid i \in \mathbb{N} \},
\]

where \(f^0(\bot) = \bot\) and \(f^{i+1}(\bot) = f(f^i(\bot))\).

(2) if \(f\) is \(\sqcap\)-continuous, then

\[
\text{gfp}(f) = \bigcap \{ f^i(\top) \mid i \in \mathbb{N} \},
\]

where \(f^0(\top) = \top\) and \(f^{i+1}(\top) = f(f^i(\top))\).
Proof

Let \((D, \leq)\) be a complete lattice and let \(f : D \rightarrow D\) be monotonic function that is \(\sqcup\)-continuous.

- We show that \(\bigsqcup\{f^i(\bot) \mid i \in \mathbb{N}\}\) is a fixpoint.

We can show that \(\{f^i(\bot) \mid i \in \mathbb{N}\}\) is an ascending chain in \(D\). Then, since the function \(f\) is \(\sqcup\)-continuous we have that

\[
f\left(\bigsqcup\{f^i(\bot) \mid i \in \mathbb{N}\}\right) = \bigsqcup\{f(f^i(\bot)) \mid i \in \mathbb{N}\}.
\]

\[
\bigsqcup\{f(f^i(\bot)) \mid i \in \mathbb{N}\} = \bigsqcup\{f^{i+1}(\bot) \mid i \in \mathbb{N}\} = \bigsqcup\{f^i(\bot) \mid i \in \mathbb{N}, i > 0\} = \bigsqcup \left(\{f^i(\bot) \mid i \in \mathbb{N}, i > 0\} \cup \{f^0(\bot)\}\right).
\]

The last equality follows from \(f^0(\bot) = \bot = \bigsqcap D\). Since \(\{f^i(\bot) \mid i \in \mathbb{N}, i > 0\} \cup \{f^0(\bot)\} = \{f^i(\bot) \mid i \in \mathbb{N}\}\), we can conclude that

\[
f\left(\bigsqcup\{f^i(\bot) \mid i \in \mathbb{N}\}\right) = \bigsqcup\{f^i(\bot) \mid i \in \mathbb{N}\}.
\]
We show that if $f(d) = d$, then $\bigcup\{f^i(\bot) \mid i \in \mathbb{N}\} \leq d$.

Let $d \in D$ be such that $f(d) = d$. It suffices to show that $f^i(\bot) \leq d$ for all $i \in \mathbb{N}$. The proof is by induction on $i \in \mathbb{N}$.

For $i = 0$ we have $f^0(\bot) = \bot \leq d$ since $\bot = \bigcap D$.

Suppose that $f^i(\bot) \leq d$ for some $i \in \mathbb{N}$. We have that

$$f^{i+1}(\bot) = f(f^i(\bot)) \leq f(d) \quad \text{(induction hypothesis and } f \text{ monotonic)}$$

$$= d \quad \text{(} d \text{ is a fixpoint)}$$
Lemma

Let \((D, \leq)\) be a complete lattice and let \(f : D \rightarrow D\) be a monotonic function.

The sequence \((f^i(\bot))_{i \in \mathbb{N}}\) is an ascending chain in \(D\).

Proof

We show by induction on \(i\) that for all \(i \in \mathbb{N}\), \(f^i(\bot) \leq f^{i+1}(\bot)\).

For \(i = 0\) we have \(f^0(\bot) = \bot \leq f^1(\bot)\), since \(\bot = \bigsqcap D\).

Suppose that \(f^i(\bot) \leq f^{i+1}(\bot)\) for some \(i \in \mathbb{N}\). We have \(f^{i+1}(\bot) = f(f^i(\bot)) \leq f(f^{i+1}(\bot)) = f^{i+2}(\bot)\). The inequality follows from the induction hypothesis and \(f\) being monotonic.
Computing least and greatest fixpoints

Theorem
Let \((D, \leq)\) be a complete lattice that satisfies (ACC) and (DCC). Let \(f : D \rightarrow D\) be a monotonic function. Then it holds that

\[
\text{lfp}(f) = \bigcup \{ f^i(\bot) \mid i \in \mathbb{N} \} \\
= f^n(\bot), \text{ where } f^n(\bot) = f^{n+1}(\bot).
\]

\[
\text{gfp}(f) = \bigcap \{ f^i(\top) \mid i \in \mathbb{N} \} \\
= f^n(\top), \text{ where } f^n(\top) = f^{n+1}(\top).
\]
Proof

Since $f$ is monotonic and $(D, \leq)$ satisfies (ACC), $f$ is \sqcup-continuous. Then, by the Theorem of Knaster, Tarski and Kleene, we have $\text{lfp}(f) = \bigsqcup \{f^i(\bot) | i \in \mathbb{N}\}$.

By the previous lemma, $\{f^i(\bot) | i \in \mathbb{N}\}$ is an ascending chain. Then, (ACC) implies that there exists $n \in \mathbb{N}$ such that for all $i \geq n$ it holds that $f^i(\bot) = f^n(\bot)$.

Take $n$ to be the first number such that $f^n(\bot) = f^{n+1}(\bot)$. We prove by induction on $i \geq n + 1$ that $f^i(\bot) = f^n(\bot)$. For $i = n + 1$ this holds by the choice of $n$. Suppose $f^i(\bot) = f^n(\bot)$. Then, we can conclude that $\bigsqcup \{f^i(\bot) | i \in \mathbb{N}\} = f^n(\bot)$. 

$\therefore$
A simple programming language: expressions

The syntax of **while-programs** is defined by the following context-free grammar defining basic expressions and programs.

**arithmetic expressions**, which denote integer values

\[
a ::= k \ | \ x \ | \ a_1 + a_2 \ | \ a_1 - a_2 \ | \ a_1 \ast a_2 \ | \ a_1/a_2
\]

\(k\) is an integer constant, \(x\) is a variable

**Boolean expressions**

\[
b ::= a_1 = a_2 \ | \ a_1 < a_2 \ | \ \neg b \ | \ b_1 \land b_2 \ | \ b_1 \lor b_2 \ | \ true
\]
A simple programming language: programs

The syntax of **while-programs** is defined by the following context-free grammar defining basic expressions and programs.

```
programs

  c ::= [skip]^l | [x := a]^l | [read(x)]^l | [print(a)]^l | c_1; c_2

  | if [b]^l then {c_1} else {c_2}

  | while [b]^l do {c}
```

*x* is a variable, *a* is an arithmetic expression, *b* is a Boolean expression

skip is a no-op statement

Statements are labelled with natural numbers: *[c]^l* where *l* ∈ \( \mathbb{N}_{>0} \). Labelled statements are called **blocks**.

All labels in a program are different.
A simple programming language

We will add other features, such as functions, later in the course.

The language lacks many features used in common programming languages: type annotations, pointers, objects, ....
Example: program

[read(x)]\(^1\);
if \([x < 0]\(^2\) then {
  \[y := 0]\(^3\);
  while \([x < 0]\(^4\) do {
    \[x := x + 1]\(^5\];
    \[y := y + 1]\(^6\]
  }
} else {
  \[y := x]\(^7\]
}
\[print(y)\]^8
Control flow graph (CFG)

A program can be represented as a finite directed graph \( G = (B, E, F) \), called control flow graph (CFG), where

- \( B \) is the set of blocks in the program
- \( E \subseteq B \) is a set of extremal (initial or final) blocks
- \( F \subseteq B \times B \) is the control flow relation

For a node \( b \in B \) we define

\[
\begin{align*}
\text{pred}(b) &= \{ b' \mid (b', b) \in F \} \quad \text{(set of predecessor nodes)} \\
\text{succ}(b) &= \{ b' \mid (b, b') \in F \} \quad \text{(set of successor nodes)}
\end{align*}
\]

We assume that

- there is at most one initial block, and it has no incoming edges
- there is at most one final block, and it has no outgoing edges

This can be ensured by adding skip statements.
Construction of the CFG

The CFG for a program can be constructed in an inductive manner.
Example: CFG

\[\text{read}(x)\] \; 
\textbf{if} \; [x < 0] \; 
\begin{align*}
[y &:= 0] \\
\textbf{while} \; [x < 0] \; \textbf{do} \; \\
[x &:= x + 1] \\
[y &:= y + 1]
\end{align*}
\textbf{else} \; 
\begin{align*}
[y &:= x]
\end{align*}
\text{print}(y)
Dataflow analysis with the monotone framework

Programs can have infinite data. Program variables have unbounded domains that do not necessarily have nice properties.

We will analyse the behaviour of programs using complete lattices as abstract data domains and monotone functions that simulate the program statements on the abstract values.

The lattice can be fixed, or can depend on the given program.

We define a set of dataflow constraints, whose solution represents information about the program. The dataflow constraints are sound if they represent correct information about the program. The analysis is conservative, since solutions may be imprecise.
Example: complete lattice of abstract values

\[ \text{read}(x) \]

\[ \text{if } x < 0 \]

\[ y := 0 \]

\[ \text{while } x < 0 \]

\[ x := x + 1 \]

\[ y := y + 1 \]

\[ \text{else} \]

\[ y := x \]

\[ \text{print}(y) \]
We are interested in definite information.

Statement $[x := x + 1]^5$

\[
\begin{align*}
    f_5(\emptyset) &= \emptyset \\
    f_5((n, d)) &= (?, d) \\
    f_5((nn, d)) &= (nn, d) \\
    f_5((?, d)) &= (?, d), \quad \text{for } d \in \{n, nn, ?\}
\end{align*}
\]
A **dataflow system** is a tuple $S = (G, (D, \leq), i, \{f_b\}_{b \in B})$, where

- $G = (B, E, F)$ is a CFG
- $(D, \leq)$ is a complete lattice that satisfies (ACC)
- $i \in D$ is an initial value for extremal blocks
- $\{f_b\}_{b \in B}$, is a family of **monotonic functions** $f_b : D \to D$, one for each block in the CFG

If we are given a complete lattice $(D, \leq)$ that satisfies (DCC), we can use the **dual lattice** $(D, \geq)$, which satisfies (ACC).
Example: \( S = (G, (D, \leq), i, \{ f_b \}_{b \in B}) \)

\[ y := 0 \]

\[ \text{while } x < 0 \text{ do } \{
    x := x + 1;
    \]

\[ y := y + 1 \]

\} \]

\[ \text{print}(y) \]

\( G = (B, E, F), E = \{1\} \)

\[ y := 0 \]

\[ \text{while } x < 0 \text{ do } \{
    x := x + 1;
    \]

\[ y := y + 1 \]

\} \]

\[ \text{print}(y) \]

\( (D, \leq) \): sign of \( y \)

\[ f_b(\emptyset) = \emptyset, \text{ for } b \in B; \text{ for } d \neq \emptyset \text{ define:} \]

\[ f_1(d) = \text{nn} \]

\[ f_2(d) = d \]

\[ f_3(d) = d \]

\[ f_4(n) = ?, f_4(\text{nn}) = \text{nn}, f_4(?) = ? \]

\[ f_5(d) = d \]
From programs to systems of equations

Let $S = (G, (D, \leq), i, \{f_b\}_{b \in B})$ be a dataflow system. We associate with each $b \in B$ a variable $X_b$ with domain $D$. $S$ defines a **system of equations** (dataflow constraints), that relate the variable at each node to those of other nodes.

For each $b \in B$, we have the following equation

$$X_b = \begin{cases} i & \text{if } b \in E \\ \bigsqcup \{f_p(X_p) \mid p \in pred(b)\} & \text{otherwise.} \end{cases}$$

- for extremal blocks: the initial value
- other blocks: the join of the values from the incoming edges
From programs to systems of equations

Let $S = (G, (D, \leq), i, \{f_b\}_{b \in B})$ be a dataflow system.
We associate with each $b \in B$ a variable $X_b$ with domain $D$.
$S$ defines a system of equations (dataflow constraints),
that relate the variable at each node to those of other nodes.

For each $b \in B$, we have the following equation

$$X_b = \begin{cases} i & \text{if } b \in E \\ \bigsqcup \{f_p(X_p) \mid p \in \text{pred}(b)\} & \text{otherwise.} \end{cases}$$

A solution of $S$ is a vector $(d_1, \ldots, d_{|B|}) \in D^{|B|}$ where for all $b \in B$

$$d_b = \begin{cases} i & \text{if } b \in E \\ \bigsqcup \{f_p(d_p) \mid p \in \text{pred}(b)\} & \text{otherwise.} \end{cases}$$
Example: solutions of system of equations

\[
y := 0^1;
\]

\[
\textbf{while } [x < 0]^2 \textbf{ do } \{
\]

\[
[x := x + 1]^3;
\]

\[
y := y + 1]^4
\]

\}

\[
\text{print}(y)^5
\]

\[
X_b = \begin{cases} 
i \\
\bigcup \{ f_p(X_p) \mid p \in \text{pred}(b) \}
\end{cases}
\]

if \( b \in E \)

otherwise.

\[
X_1 = ?
\]

\[
X_2 = f_1(X_1) \sqcup f_4(X_4)
\]

\[
X_3 = f_2(X_2)
\]

\[
X_4 = f_3(X_3)
\]

\[
X_5 = f_2(X_2)
\]
Example: solutions of system of equations

\[ y := 0 \]
\[ \textbf{while } [x < 0] \textbf{ do } \{
\[ x := x + 1 \]
\[ y := y + 1 \]
\} \]
\[ \textbf{print}(y) \]

\[ X_b = \begin{cases} 
  i & \text{if } b \in E \\
  \bigcup \{ f_p(X_p) \mid p \in \text{pred}(b) \} & \text{otherwise.}
\end{cases} \]

\[ X_1 = ? \quad i = ? \]
\[ X_2 = f_1(X_1) \sqcup f_4(X_4) \quad f_1(d) = \text{nn} \]
\[ X_3 = f_2(X_2) \quad f_2(d) = d \]
\[ X_4 = f_3(X_3) \quad f_3(d) = d \]
\[ X_5 = f_2(X_2) \quad f_4(n) = ?, f_4(\text{nn}) = \text{nn}, f_4(?) = ? \]
\[ f_5(d) = d \]

The vector \((?, \text{nn}, \text{nn}, \text{nn}, \text{nn})\) is a solution.
Solutions as fixpoints

Let $S = (G, (D, \leq), i, \{f_b\}_{b \in B})$ be a dataflow system.

Consider the lattice $(D_S, \leq_S)$, where $D_S = D^{\mid B\mid}$ and
$(d_1, \ldots, d_{\mid B\mid}) \leq_S (d'_1, \ldots, d'_{\mid B\mid})$ if and only if $d_b \leq d'_b$ for all $b \in B$.
Define the function $g_S : D_S \to D_S$ such that
$g_S((d_1, \ldots, d_{\mid B\mid})) = (d'_1, \ldots, d'_{\mid B\mid})$ where for all $b \in B$

\[
\begin{align*}
    d'_b &= \begin{cases} 
        i & \text{if } b \in E \\
        \bigsqcup \{f_p(d_p) \mid p \in \text{pred}(b)\} & \text{otherwise.}
    \end{cases}
\end{align*}
\]

Recall

\[
X_b = \begin{cases} 
        i & \text{if } b \in E \\
        \bigsqcup \{f_p(X_p) \mid p \in \text{pred}(b)\} & \text{otherwise.}
    \end{cases}
\]

Then, a vector $\overline{d} = (d_1, \ldots, d_{\mid B\mid}) \in D^{\mid B\mid}$ is a solution of $S$ if and only if $\overline{d}$ is a fixpoint of $g_S$, that is $g_S(\overline{d}) = \overline{d}$. 

Example: fixpoint computation

\[ y := 0 \]
\[ \text{while } [x < 0] \text{ do } \{
\]
\[ [x := x + 1];
\[ [y := y + 1]
\}
\[ \text{print}(y) \]

\[ d'_b = \begin{cases} 
  i & \text{if } b \in E \\
  \bigsqcup \{ f_p(d_p) \mid p \in \text{pred}(b) \} & \text{otherwise.}
\end{cases} \]

\[
\begin{align*}
  i &= ? \\
  f_1(d) &= \text{nn} \\
  f_2(d) &= d \\
  f_3(d) &= d \\
  f_4(n) &= ? \\
  f_4(nn) &= \text{nn} \\
  f_4(?) &= ? \\
  f_5(d) &= d
\end{align*}
\]
Example: fixpoint computation

\[ y := 0 \]

while \( x < 0 \) do {

\[ x := x + 1 \]

\[ y := y + 1 \]

} 

[print(y)]

\[ d'_b = \begin{cases} 
  i & \text{if } b \in E \\
  \bigsqcup \{ f_p(d_p) \mid p \in \text{pred}(b) \} & \text{otherwise.}
\end{cases} \]

\[ g_S((\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)) = (?, \emptyset, \emptyset, \emptyset, \emptyset) \]
Example: fixpoint computation

\[ y := 0 \]

while \([x < 0]\) do {
  \[ x := x + 1 \]
  \[ y := y + 1 \]
}

\[ \text{print}(y) \]

\[ d'_b = \begin{cases} 
  i & \text{if } b \in E \\
  \bigsqcup \{ f_p(d_p) \mid p \in \text{pred}(b) \} & \text{otherwise.}
\end{cases} \]

\[
\begin{align*}
  i &= \text{?} \\
  f_1(d) &= \text{nn} \\
  f_2(d) &= d \\
  f_3(d) &= d \\
  f_4(n) &= \text{?} \\
  f_4(nn) &= \text{nn} \\
  f_4(?) &= \text{?} \\
  f_5(d) &= d
\end{align*}
\]

\[ g_S((\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)) = (\text{?}, \emptyset, \emptyset, \emptyset, \emptyset) \]

\[ g_S((\text{?}, \emptyset, \emptyset, \emptyset, \emptyset)) = (\text{?}, \text{nn}, \emptyset, \emptyset, \emptyset) \]
Example: fixpoint computation

\[
[y := 0];
\]

while \( [x < 0] \) do {
\[
[x := x + 1];
[y := y + 1]
\]
}

\( [\text{print}(y)] \)

\[
d'_b = \begin{cases} 
  i & \text{if } b \in E \\
  \bigcup \{ f_p(d_p) \mid p \in \text{pred}(b) \} & \text{otherwise.}
\end{cases}
\]

\[
i = ?
\]
\[
f_1(d) = nn
\]
\[
f_2(d) = d
\]
\[
f_3(d) = d
\]
\[
f_4(n) = ?
\]
\[
f_4(nn) = nn
\]
\[
f_4(?) = ?
\]
\[
f_5(d) = d
\]

\[
g_S((\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)) = (?, \emptyset, \emptyset, \emptyset, \emptyset)
\]
\[
g_S((?, \emptyset, \emptyset, \emptyset, \emptyset)) = (?, nn, \emptyset, \emptyset, \emptyset)
\]
\[
g_S((?, nn, \emptyset, \emptyset, \emptyset)) = (?, nn, nn, \emptyset, nn)
\]

\[
g_S((?, nn, nn, \emptyset, nn)) = (?, nn, nn, nn, nn)
\]
Example: fixpoint computation

\[ y := 0 \]

while \( x < 0 \) do {

\[ x := x + 1 \]

\[ y := y + 1 \]

}\n
[print(y)]

\[ d'_{b} = \begin{cases} i & \text{if } b \in E \\
\bigcup \{ f_{p}(d_{p}) | p \in \text{pred}(b) \} & \text{otherwise.} \end{cases} \]

\begin{align*}
i & = ? \\
f_{1}(d) & = \text{nn} \\
f_{2}(d) & = d \\
f_{3}(d) & = d \\
f_{4}(n) & = ? \\
f_{4}(\text{nn}) & = \text{nn} \\
f_{4}(?) & = ? \\
f_{5}(d) & = d \\
g_{S}(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset) & = (?, \emptyset, \emptyset, \emptyset, \emptyset) \\
g_{S}(?, \emptyset, \emptyset, \emptyset, \emptyset) & = (?, \text{nn}, \emptyset, \emptyset, \emptyset) \\
g_{S}(?, \text{nn}, \emptyset, \emptyset, \emptyset) & = (?, \text{nn}, \text{nn}, \emptyset, \text{nn}) \\
g_{S}(?, \text{nn}, \text{nn}, \emptyset, \text{nn}) & = (?, \text{nn}, \text{nn}, \text{nn}, \text{nn})
\end{align*}
Example: fixpoint computation

\[ y := 0^1; \]
while \[ x < 0^2 \] do \{
\[ x := x + 1^3; \]
\[ y := y + 1^4 \]
\}

\[ \text{print}(y)^5 \]

\[ d'_b = \begin{cases} 
    i & \text{if } b \in E \\
    \bigcup \{ f_p(d_p) \mid p \in \text{pred}(b) \} & \text{otherwise.}
\end{cases} \]

\[ \begin{align*}
    i & = ? \\
    f_1(d) & = nn \\
    f_2(d) & = d \\
    f_3(d) & = d \\
    f_4(n) & = ? \\
    f_4(nn) & = nn \\
    f_4(?) & = ? \\
    f_5(d) & = d
\end{align*} \]

\[ \begin{align*}
    g_S((\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)) & = (?, \emptyset, \emptyset, \emptyset, \emptyset) \\
    g_S((?, \emptyset, \emptyset, \emptyset, \emptyset)) & = (?, nn, \emptyset, \emptyset, \emptyset) \\
    g_S((?, nn, \emptyset, \emptyset, \emptyset)) & = (?, nn, nn, \emptyset, nn) \\
    g_S((?, nn, nn, \emptyset, nn)) & = (?, nn, nn, nn, nn) \\
    g_S((?, nn, nn, nn, nn)) & = (?, nn, nn, nn, nn)
\end{align*} \]