Addendum to the Article “Types vs. PDGs in Information Flow Analysis” – Proofs and Operational Semantics

Heiko Mantel and Henning Sudbrock

Computer Science Department, TU Darmstadt, Germany
{mantel,sudbrock}@mais.informatik.tu-darmstadt.de

This document contains proofs for theorems from the article “Types vs. PDGs in Information Flow Analysis” [MS13] (in Sections 1 and 2). Moreover as an addendum to the article it contains the operational semantics for the considered programming language (in Section 3).

1 Proof of Lemma 1

Before proving Lemma 1 from [MS13] we prove several propositions that relate paths in the graph PDG(CFG\{I\},\{O\}) where c is of the form if(e) then c_1 else c_2 fi, while(e) do c_1 od, or c_1;c_2 to paths in the graphs PDG(CFG\{I\},\{O\}) and (if applicable) PDG(CFG\{I\},\{O\}). In the proofs, we write p+k for the path that is obtained from p by adding k to each node on p that is a natural number, and leaving start and stop unchanged. Moreover, we write p−k for the path that is obtained from p by subtracting k from each node on p that is a natural number, and leaving start and stop unchanged.

Proposition 1. Let c = if(e) then c_1 else c_2 fi and x, y ∈ Var. Then the following hold:

1. There is a path from in to out in PDG(CFG\{x\},\{y\}) that contains more than 2 nodes if and only if one of the following conditions are satisfied:
   (a) there is a path from in to out in PDG(CFG\{x\},\{y\}),
   (b) there is a path from in to out in PDG(CFG\{y\},\{x\}),
   (c) x ∈ fv(e) and there is a path from start to out in PDG(CFG\{x\},\{y\}) that contains more than 2 nodes, or
   (d) x ∈ fv(e) and there is a path from start to out in PDG(CFG\{y\},\{x\}) that contains more than 2 nodes.

2. There is a path from start to out in PDG(CFG\{x\},\{y\}) that contains more than 2 nodes if and only there is such a path in PDG(CFG\{x\},\{y\}) or in PDG(CFG\{y\},\{x\}).

Proof. We firstly prove Statement (1) of the proposition.

1. By the construction of the CFG for commands the following holds:
   (a) Let n, n' \notin \{1, start, stop\} be nodes of CFG. Then n' is data (control) dependent on n for CFG if and only if n' ∈ 1 is data (control) dependent on
7. Assume finally that \( x \) of the first statement of the proposition, paths from Node \( \text{start} \) to Node \( \text{stop} \) is seen as follows: Using (1) and (2) of the proof of the first statement of the proposition, paths from Node \( \text{start} \) to Node \( \text{out} \) in

\[ n \odot 1 \text{ for } CFG_{c_1} \text{ or } n' \odot 1 \odot |c_1| \text{ is data (control) dependent on } n \odot 1 \odot |c_1| \text{ for } CFG_{c_2}. \]

Moreover, \( n \) is not data dependent on 1 and 1 is not data dependent on \( n \). Moreover, \( n \) is control dependent on 1 if and only if \( n \odot 1 \) is control dependent on \( \text{start} \) for \( CFG_{c_1} \) or \( n \odot 1 \odot |c_1| \) is control dependent on \( \text{start} \) for \( CFG_{c_2} \).

2. By the construction of the PDG and the construction of the CFG for commands we obtain the following:

(a) Let \( n \not\in \{1, \text{start}, \text{stop}\} \) be a node of \( CFG_c \). Then there is an edge \( (in, n) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) if and only if there is an edge \( (in, n \odot 1) \) in the graph \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or an edge \( (in, n \odot 1 |c_1|) \) in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

Moreover, there is an edge \( (n, out) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) if and only if there is an edge \( (n \odot 1, out) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or an edge \( (n \odot 1 \odot |c_1|, out) \) in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

(b) There is an edge \( (in, 1) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) if and only if \( x \in fv(e) \).

Moreover, there is no edge \( (1, out) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \).

(c) There is an edge \( (in, out) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) if and only if there is an edge \( (in, out) \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

3. Assume that there is a path \( p \) from \( in \) to \( out \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \).

(a) If \( p = (in, out) \), then by (2a) \( p \) is also a path in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

(b) If \( p = (in, 1), p' \) for some \( p' \), then, by (2b), \( x \in fv(e) \) and \( p' \neq (out) \). Moreover, by (1a) the first node in \( p' \) is control dependent on \( \text{start} \) in \( CFG_c \), and, since due to (1a) all edges in \( p' \) derive from edges in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or \( PDG(CFG_{c_2}^{(x)}, \{y\}) \), \( (\text{start}, p' \odot 1) \) is a path from \( \text{start} \) to \( out \) that contains more than 2 nodes in the graph \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or in the graph \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

(c) If \( p \neq (in, out) \) and \( p \neq (in, 1), p' \) for some \( p' \), then, using (1a) and (2a), \( p - 1 \) is a path in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \) or in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

4. Assume that there is a path \( p \) from \( in \) to \( out \) in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \). Then, by (1a), (2a), and (2c), \( p + 1 \) is a path in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \).

5. Assume that there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \). Then, by (1a), (2a), and (2c), \( p + (1 + |c_1|) \) is a path in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

6. Assume that \( x \in fv(e) \) and that there is a path from \( \text{start} \) to \( out \) in the graph \( PDG(CFG_{c_1}^{(x)}, \{y\}) \), i.e., of the form \( \langle \text{start}, p' \rangle \). Then, by (1a), (2a), and (2b), the sequence \( \langle in, 1 \rangle, (p' + 1) \) is a path in \( PDG(CFG_{c_1}^{(x)}, \{y\}) \).

7. Assume finally that \( x \in fv(e) \) and that there is a path from \( \text{start} \) to \( out \) in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \) that contains more than 2 nodes, i.e., of the form \( \langle \text{start}, p' \rangle \). Then, by (1a), (2a), and (2b), the sequence \( \langle in, 1 \rangle, (p' + (1 + |c_1|)) \) is a path in \( PDG(CFG_{c_2}^{(x)}, \{y\}) \).

Statement (2) of the proposition is seen as follows: Using (1) and (2) of the proof of the first statement of the proposition, paths from Node \( \text{start} \) to Node \( \text{out} \) in
Proposition 2. Let \( c = c_1; c_2 \) and \( x, y \in \text{Var} \). Then the following hold:

1. There is a path from in to out in \( \text{PDG}(\text{CFG}^{(x)}_{c_1};\{y\}) \) if and only if there exists \( z \in \text{Var} \) such that there is a path from in to out in \( \text{PDG}(\text{CFG}^{(x)}_{c_1};\{z\}) \) and a path from in to out in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \).

2. There is a path from start to out in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \) that contains more than 2 nodes if and only if one of the following conditions is satisfied:
   (a) there exists \( z \in \text{Var} \) such that there is a path from start to out that contains more than 2 nodes in the graph \( \text{PDG}(\text{CFG}^{(x)}_{c_1};\{z\}) \) and a path from start to out in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \), or
   (b) there is a path from start to out in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \) that contains more than 2 nodes.

Proof. We firstly prove Statement (1).

1. By the construction of the CFG for commands, the following hold:
   (a) Let \( n, n' \not\in \{\text{start, stop}\} \) be nodes of \( \text{CFG}_c \). Then \( n' \) is control dependent on \( n \) for \( \text{CFG}_c \), if and only if \( n' \) is control dependent on \( n \) for \( \text{CFG}_{c_1} \), or if \( n \odot |c_1| \) is control dependent on \( n \odot |c_1| \) for \( \text{CFG}_{c_2} \).
   (b) Let \( n, n' \not\in \{\text{start, stop}\} \) be nodes of \( \text{CFG}_c \). Then \( n' \) is data dependent on \( n \) for \( \text{CFG}_c \), if and only if one of the following conditions is satisfied:
      i. \( n' \) is data dependent on \( n \) for \( \text{CFG}_{c_1} \)
      ii. \( n' \odot |c_1| \) is data dependent on \( n \odot |c_1| \) for \( \text{CFG}_{c_2} \)
      iii. There exists \( z \in \text{Var} \) such that \( z \in \text{def}_{c_1}(n) \), \( z \in \text{use}_{c_2}(n' \odot |c_1|) \), a definition of \( z \) at \( n \) reaches \( \text{stop} \) in \( \text{CFG}_{c_1} \), and a definition of \( z \) at \( \text{start} \) reaches \( n' \odot |c_1| \) in \( \text{CFG}_{c_2} \).

2. By the definition of PDGs and the definition of CFGs of commands there is an edge \( (in, out) \) in \( \text{PDG}(\text{CFG}^{(x)}_{c_1};\{y\}) \) if and only if \( x = y \) and \( (in, out) \) is an edge both in \( \text{PDG}(\text{CFG}^{(z)}_{c_1};\{y\}) \) and in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \).

3. Assume that there is a path \( p \) from \( in \) to \( out \) in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \).
   (a) If \( p = (in, out) \), then, by (2), we conclude (setting \( z = x = y \)).
   (b) If \( p \) contains, besides \( in \) and \( out \), only nodes in the set \( \{1, \ldots, |c_1|\} \), then a definition of \( y \) at the one but last node of \( p \) reaches \( \text{stop} \) in \( \text{CFG}_c \), and, hence, a definition of \( y \) at \( \text{start} \) reaches \( \text{stop} \) in \( \text{CFG}_{c_2} \). Hence, \( p \) is a path in \( \text{PDG}(\text{CFG}^{(z)}_{c_2};\{y\}) \) and \( (in, out) \) is a path in \( \text{PDG}(\text{CFG}^{(y)}_{c_2};\{y\}) \). Thus, we conclude setting \( z = y \).
(c) If \( p \) contains, besides \( \text{in} \) and \( \text{out} \), only nodes in \( \{|c_1|+1, \ldots, |c_1|+2\} \), we argue as in the previous case with roles of \( c_1 \) and \( c_2 \) switched, concluding by setting \( z = x \).

(d) If \( p \) contains nodes in both \( \{1, \ldots, |c_1|\} \) and \( \{|c_1|+1, \ldots, |c_1|+2\} \), then, by the definition of the CFG, \( p = p_1.p_2 \) where \( p_1 \) contains only nodes in \( \{1, \ldots, |c_1|\} \) and \( p_2 \) contains only nodes in \( \{|c_1|+1, \ldots, |c_1|+2\} \). With (1b.iii) it follows that there is \( z \) such that \( p_1.(\text{out}) \) is a path in the graph \( \text{PDG}(CFG_{c_1}^{(x)},(z)) \) and \( (\text{in}).p_2 \) is a path in \( \text{PDG}(CFG_{c_2}^{(y)},(y)) \).

4. Now assume that there exists \( z \in \text{Var} \) and paths \( p_1 \) and \( p_2 \) from \( \text{in} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_1}^{(x)},(z)) \) and in \( \text{PDG}(CFG_{c_2}^{(y)},(y)) \), respectively. With (1b.iii) it follows that there exists a path from \( \text{in} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c}^{(z)},(z)) \) (by joining these two paths).

We now prove Statement (2).

1. By the construction of the CFG for \( c_1; c_2 \) and the definition of postdominance, a node \( n \) is control dependent on \( \text{start} \) in \( CFG_c \) if and only if \( n \) is control dependent on \( \text{start} \) in \( CFG_{c_1} \) or \( n \in |c_1| \) is control dependent on \( \text{start} \) in \( CFG_{c_2} \).

2. Assume that \( p \) is a path from \( \text{start} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_1}^{(x)},(y)) \) that contains more than 2 nodes.

(a) If \( p \) contains, besides \( \text{start} \) and \( \text{out} \), only nodes in \( \{|c_1|+1, \ldots, |c_1|+2\} \), then \( p - |c_1| \) is a path from \( \text{start} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_2}^{(y)},(y)) \) that contains more than 2 nodes (by (1), (1a), (1b), and the proof of the first statement of the proposition).

(b) Otherwise, using (1) and arguing analogously to the proof of the first part of the proposition, there exists \( z \) such that there are paths from \( \text{start} \) to \( \text{out} \) in the graph \( \text{PDG}(CFG_{c_1}^{(x)},(z)) \) (containing more than 2 nodes) and from \( \text{in} \) to \( \text{out} \) in the graph \( \text{PDG}(CFG_{c_2}^{(y)},(y)) \).

3. The backwards direction (assuming paths in the graphs \( \text{PDG}(CFG_{c_1}^{(x)},(y)) \) and \( \text{PDG}(CFG_{c_2}^{(y)},(y)) \)) is analogous to the previous cases.

**Proposition 3.** Let \( c = \text{while}(c_1) \). Then the following hold:

1. There is a path from \( \text{in} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c}^{(x)},(y)) \) if and only if there exist \( z_1, \ldots, z_k \) (for some \( k > 1 \)) with \( z_1 = x \) and \( z_k = y \) such that for each \( i \in \{1, \ldots, k-1\} \) one of the following conditions is satisfied:
   (a) there is a path from \( \text{in} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_1}^{(z_i)},(z_{i+1})) \), or
   (b) \( z_i \in fv(e) \) and there is a path from \( \text{start} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_1}^{(z_i)},(z_{i+1})) \) that contains more than 2 nodes.

2. There is a path from \( \text{start} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c}^{(x)},(y)) \) that contains more than 2 nodes if and only if there exist \( z_1, \ldots, z_k \) (for some \( k > 0 \)) such that \( z_k = y \), there is a path from \( \text{start} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_1}^{(z_1)},(z_2)) \) that contains more than 2 nodes, and for each \( i \in \{1, \ldots, k-1\} \) one of the following conditions is satisfied:
   (a) there is a path from \( \text{in} \) to \( \text{out} \) in \( \text{PDG}(CFG_{c_1}^{(z_i)},(z_{i+1})) \), or
(b) $z_i \in \text{fv}(e)$ and there is a path from start to out in $\text{PDG}(\text{CFG}_{c_1}^{[z_i]}\{z_{i+1}\})$ that contains more than 2 nodes.

**Proof.** We start proving Statement (1) of the proposition.

1. By the construction of the CFG for commands the following hold:
   (a) Let $n, n' \notin \{1, \text{start}, \text{stop}\}$ be nodes of $\text{CFG}_c$. Then $n'$ is data dependent on $n$ for $\text{CFG}_c$ if and only if one of the following holds:
      i. $n' \notin 1$ is data dependent on $n \in 1$ for $\text{CFG}_{c_1}$ or
      ii. there exists $z \in \text{Var}$ such that $(n \subset 1, \text{out})$ is an edge in the graph $\text{PDG}(\text{CFG}_{c_1}^{[x]}\{z\})$ and $(in, n' \subset 1)$ is an edge in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.
   Moreover, Node 1 is data dependent on $n$ if and only if there exists $z \in \text{fv}(e)$ such that $(n, \text{out})$ is an edge in $\text{PDG}(\text{CFG}_c^{[x]}\{z\})$.
   (b) Let $n, n' \notin \{1, \text{start}, \text{stop}\}$ be nodes of $\text{CFG}_c$. Then $n'$ is control dependent on $n$ for $\text{CFG}_c$ if and only if $n' \subset 1$ is control dependent on $n \subset 1$ for $\text{CFG}_{c_1}$.
   Moreover, $n$ is control dependent on 1 if and only if $n \subset 1$ is control dependent on start for $\text{CFG}_{c_1}$.

2. By the construction of the PDG and the construction of the CFG for commands the following hold:
   (a) Let $n \notin \{1, \text{start}, \text{stop}\}$ be a node of $\text{CFG}_c$. Then there is an edge $(in, n)$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$ if and only if there is an edge $(in, n \subset 1)$ in the graph $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.
   Moreover, there is an edge $(n, \text{out})$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$ if and only if there is an edge $(n \subset 1, \text{out})$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.
   (b) There is an edge $(in, 1)$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$ if and only if $x \in \text{fv}(e)$. Moreover, there is no edge $(1, \text{out})$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.
   (c) There is an edge $(in, \text{out})$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$ if and only if there is an edge $(in, \text{out})$ in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.

3. Assume that there is a path $p$ from in to out in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.
   (a) If $p = (in, \text{out})$, then by (2c) it is also a path in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$.
   (b) If $p$ consists of more than 2 nodes, then by (1) and (2) $p$ consists of segments that correspond to paths in $\text{PDG}(\text{CFG}_c^{[x]}\{y\})$, potentially separated by Node 1 (representing the guard of the loop), where the edges $(n, n')$ between these segments correspond to edges $(n, \text{out})$ in $\text{PDG}(\text{CFG}_{c_1}^{[x]}\{z_{i+1}\})$ and $(in, n')$ respectively (start, $n'$) in $\text{PDG}(\text{CFG}_{c_1}^{[x]}\{z_{i+1}\})$ for a sequence of $z_i$.
   (c) Hence, there exist $z_1, \ldots, z_k$ and paths from in to out respectively from start to out (containing more than 2 nodes when starting an start) in the PDGs $\text{PDG}(\text{CFG}_{c_1}^{[x]}\{z_{i+1}\})$, where $z_1 = x$ and $z_k = y$.

4. Assume now that there exist $z_1, \ldots, z_k$ with $z_1 = x$ and $z_k = y$ and paths from in to out respectively from start to out (containing more than 2 nodes) in the PDGs $\text{PDG}(\text{CFG}_{c_1}^{[x]}\{z_{i+1}\})$.
   (a) Using (1) and (2), these paths can be concatenated using the same arguments as in Propositions 1 and 2, where Node start is replaced by Node 1.
Statement (2) of the proposition is proven analogously to Statement (1), with the only difference being that the first segment of the path now begins at Node start, not at Node in.

We prove a generalization of Lemma 1 from [MS13] that permits arbitrary security domains for the program counter in the typing judgment. The generalization permits an inductive proof over the construction of the command \( c \).

**Proposition 4.** Let \( c \in \text{Com} \), \( \Gamma : \text{Var} \rightarrow \mathcal{D} \), \( pc \in \mathcal{D} \), and \( y \in \text{Var} \). Let \( \Gamma' \) be the environment with \( pc \vdash \Gamma(c) \). Let \( X \subseteq \text{Var} \) be such that \( x \in X \) if and only if there is a path \( \langle \text{in}, \ldots, \text{out} \rangle \) in \( \text{PDG}(\text{CFG}_x^{\langle y \rangle}) \). Then one of the following two conditions is satisfied:

1. \( \Gamma'(y) = pc \cup (\bigcup_{x \in X} \Gamma(x)) \), and there is a path from start to out in the graph \( \text{PDG}(\text{CFG}_x^{\langle y \rangle}) \) for some \( x \in \text{Var} \) that contains more than 2 nodes.
2. \( \Gamma'(y) = \bigcup_{x \in X} \Gamma(x) \), and there is no such path in \( \text{PDG}(\text{CFG}_x^{\langle y \rangle}) \).

**Proof (Proof of Proposition 4).** The proof is by induction on the structure of \( c \).

**Assume that** \( c = \text{skip} \):

1. By the typing rule \([\text{skip}]\), \( \Gamma = \Gamma' \). Hence, \( \Gamma(y) = \Gamma'(y) \).
2. The node start is the only node in \( \text{CFG}_{\text{skip}} \) with two outgoing edges, and, hence, all control dependency edges in \( \text{PDG}(\text{CFG}_{\text{skip}}^{\langle y \rangle}) \) originate at start.
3. Node 1 (representing the skip-statement) has empty def and use sets. Hence, \( \text{PDG}(\text{CFG}_{\text{skip}}^{\langle y \rangle}) \) contains no data dependency edges from or to Node 1, and it contains an edge \( \langle \text{in}, \text{out} \rangle \) if and only if \( x = y \).
4. By (2) and (3), there is a path from \text{in} to \text{out} in \( \text{PDG}(\text{CFG}_{\text{skip}}^{\langle y \rangle}) \) if and only if \( x = y \), i.e., \( X = \{ y \} \).
5. Moreover, by (2) and (3) there is no path from start to out in the graph \( \text{PDG}(\text{CFG}_{\text{skip}}^{\langle y \rangle}) \) that contains more than 2 nodes.
6. Hence, by (4) and (5), it suffices to show that \( \Gamma'(y) = \Gamma(y) \) to conclude this case, and \( \Gamma'(y) = \Gamma(y) \) holds by (1).

**Assume that** \( c = z := e \):

1. By the same argument as in the case for \text{skip}, all control dependency edges in \( \text{PDG}(\text{CFG}_{\text{skip}}^{\langle y \rangle}) \) originate at start.
2. Be the definition of def and use sets, \( \text{def}_c(1) = \{ z \} \) and \( \text{use}_c(1) = \text{fv}(e) \) (where 1 is the node representing the assignment). Hence, by the definition of data dependence there is a data dependency edge \( \langle \text{in}, 1 \rangle \) in \( \text{PDG}(\text{CFG}_{\text{skip}}^{\langle y \rangle}) \) if and only if \( x \in \text{fv}(e) \) and there is a data dependency edge \( \langle 1, \text{out} \rangle \) if and only if \( y = z \). Moreover, there is an edge \( \langle \text{in}, \text{out} \rangle \) if and only if \( x = y \) and \( x \neq z \).
3. Assume that \( y = z \).
   (a) By (1) and (2), there is a path from \text{in} to \text{out} if and only if \( x \in \text{fv}(e) \) (the path \( \langle \text{in}, 1, \text{out} \rangle \)). Hence, \( X = \text{fv}(e) \).
   (b) Since node 1 is control dependent on start there is a path from start to \text{out} that contains more than 2 nodes (the path \( \langle \text{start}, 1, \text{out} \rangle \)).
(c) Hence, we must show that \( \Gamma'(y) = pc \cup (\bigcup_{x \in f(u)} \Gamma(x)) \). This equality holds due to the typing rules \([\text{assign}]\) and \([\text{exp}]\).

4. Assume now that \( y \neq z \).
   
   (a) By (1) and (2), there is a path from \( in \) to \( out \) if and only if \( x = y \) and \( x \neq z \) (the path \( \langle in, out \rangle \)). Hence, \( X = \{y\} \).
   
   (b) Moreover, by (1) and (2) there is no path from \( start \) to \( out \) that contains more than 2 nodes.
   
   (c) Hence, we must show that \( \Gamma'(y) = \Gamma(y) \). This holds due to the typing rule \([\text{assign}]\).

**Assume that** \( c = c_1;c_2 \):

1. By typing rule \([\text{seq}]\) there is a domain environment \( \Gamma'' \) such that \( pc \vdash \Gamma\{c_1\}\Gamma'' \) and \( pc \vdash \Gamma''\{c_2\} \Gamma'' \).

2. Let \( X_2 \) be the set of variables such that \( x \in X_2 \) if and only if there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_2}^{(x)};\{y\}) \). By the induction hypothesis for \( k \) and \( c_2 \) (instantiating the environment with \( \Gamma'' \)) one of the following conditions is satisfied:
   
   (a) \( \Gamma'(y) = pc \cup (\bigcup_{x \in X_2} \Gamma''(x)) \) and there is a path from \( start \) to \( out \) in the graph \( PDG(CFG_{c_2}^{(x)};\{y\}) \) that contains more than 2 nodes, or
   
   (b) \( \Gamma'(y) = \bigcup_{x \in X_2} \Gamma''(x) \), and there is no such path in \( PDG(CFG_{c_2}^{(x)};\{y\}) \).

3. For each \( z \in X_2 \), let \( X_z \) be the set of variables such that \( x \in X_z \) if and only if there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_2}^{(x)};\{z\}) \). For each \( z \in X_2 \), by the induction hypothesis for \( c_1 \) (instantiating the environment with \( \Gamma \) and the variable with \( z \)) one of the following conditions is satisfied:
   
   (a) \( \Gamma''(z) = pc \cup (\bigcup_{x \in X_z} \Gamma(x)) \) and there is a path from \( start \) to \( out \) in the graph \( PDG(CFG_{c_1}^{(x)};\{z\}) \) that contains more than 2 nodes, or
   
   (b) \( \Gamma''(z) = \bigcup_{x \in X_z} \Gamma(x) \), and there is no such path in \( PDG(CFG_{c_1}^{(x)};\{z\}) \).

4. By Proposition 2, there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_1}^{(x)};\{y\}) \) if and only if there exists \( z \in \text{Var} \) such that there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_1}^{(x)};\{z\}) \) and a path from \( in \) to \( out \) in \( PDG(CFG_{c_1}^{(x)};\{y\}) \). Hence, it follows from the definitions of \( X_2 \) and \( X_z \) that \( X = \bigcup_{x \in X_2} X_z \).

5. We distinguish the cases (2a) and (2b). Assume firstly that (2a) holds.
   
   (a) Due to (2a) and Proposition 2, for all \( z \in \text{Var} \) there is a path from \( start \) to \( out \) in \( PDG(CFG_{c_1;c_2}^{(x)};\{y\}) \) that contains more than 2 nodes. Hence, we must show that \( \Gamma'(y) = pc \cup (\bigcup_{x \in X} \Gamma(x)) \).
   
   (b) Due to (2a), \( \Gamma'(y) = pc \cup (\bigcup_{x \in X_2} \Gamma''(x)) \).
   
   (c) Due to (3), for each \( z \in X_2 \) either \( \Gamma''(z) = pc \cup (\bigcup_{x \in X_z} \Gamma(x)) \) or \( \Gamma''(z) = (\bigcup_{x \in X_z} \Gamma(x)) \) holds.
   
   (d) It follows from (b) and (c) that \( \Gamma'(y) = pc \cup (\bigcup_{x \in X_2} \bigcup_{x \in X_z} \Gamma(x)) \). Hence, by (4), \( \Gamma'(y) = pc \cup (\bigcup_{x \in X} \Gamma(x)) \).

6. Assume now that (2b) holds.
   
   (a) Due to (2b), \( \Gamma'(y) = \bigcup_{x \in X_2} \Gamma''(x) \).
   
   (b) For each \( z \in X_2 \), either (3a) or (3b) holds. We distinguish the cases that (3b) holds for all \( z \in X_2 \) and that (3b) does not hold for some \( z \in X_2 \).
(c) Assume firstly that (3b) holds for all \( z \in X_2 \).
   i. By Proposition 2, \( PDG(CFG_{c_1; c_2}^{(x)}; \{y\}) \) does not contain a path from \( start \) to \( out \) that contains more than 2 nodes.
   ii. For all \( z \in X_2 \), it follows from (3b) that \( \Gamma''(z) = (\bigcup_{x \in X_2} \Gamma(x)) \).
   iii. From (a) and (ii) it follows that \( \Gamma''(y) = \bigcup_{x \in X_2} \bigcup_{x \in X} \Gamma(x) \). Hence, by (4), \( \Gamma''(y) = \bigcup_{x \in X} \Gamma(x) \).

(d) Assume now that there exists \( z \in X_2 \) such that (3a) holds for \( z \).
   i. Hence, \( \Gamma''(z) = pc \cup (\bigcup_{x \in X_2} \Gamma(x)) \).
   ii. Moreover, \( PDG(CFG_{c_1; c_2}^{(x)}; \{z\}) \) contains a path from \( start \) to \( out \) that contains more than 2 nodes.
   iii. Since \( z \in X_2 \) there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_2}^{(z)}; \{y\}) \).
   iv. By Proposition 2, (ii), and (iii), \( PDG(CFG_{c_1; c_2}^{(x)}; \{y\}) \) contains a path from \( start \) to \( out \) that contains more than 2 nodes.
   v. Due to (3), for each \( z \in X_2 \) either \( \Gamma''(z) = pc \cup (\bigcup_{x \in X_2} \Gamma(x)) \) or \( \Gamma''(z) = (\bigcup_{x \in X_2} \Gamma(x)) \) holds.
   vi. From (a), (i), and (v), it follows that \( \Gamma''(y) = pc \bigcup (\bigcup_{x \in X_2} \bigcup_{x \in X} \Gamma(x)) \). Hence, by (3), \( \Gamma''(y) = pc \bigcup (\bigcup_{x \in X} \Gamma(x)) \).

Assume that \( c = \text{if} (e) \) then \( c_1 \) else \( c_2 \) fi:

1. Let \( \{z_1, \ldots, z_l\} = fv(e) \), and \( t = dom(z_1) \cup \ldots \cup dom(z_l) \).
2. By the typing rule \([e]\), there are environments \( \Gamma_1' \) and \( \Gamma_2' \) such that \( \Gamma'' = \Gamma_1' \sqcup \Gamma_2' \), \( pc \sqcup t \vdash \Gamma \{c_1\} \Gamma_1' \), and \( pc \sqcup t \vdash \Gamma \{c_2\} \Gamma_2' \).
3. For \( i \in \{1, 2\} \), let \( X_i \) be the set of variables such that \( x \in X_i \) if and only if there is a path from \( in \) to \( out \) in \( PDG(CFG_{c_1}^{(x)}; \{y\}) \).
4. Applying the induction hypothesis for both \( k \) and \( c_1 \) and \( k \) and \( c_2 \) (instantiating the program counter security level with \( pc \sqcup t \)), it follows that for \( i \in \{1, 2\} \) one of the following two conditions is satisfied:
   a. \( \Gamma''(y) = pc \sqcup t \sqcup (\bigcup_{x \in X_2} \Gamma(x)) \) and there is a path from \( start \) to \( out \) in the graph \( PDG(CFG_{c_1}^{(x)}; \{y\}) \) that contains more than 2 nodes, or
   b. \( \Gamma''(y) = \bigcup_{x \in X_2} \Gamma(x) \) and there is no such path in \( PDG(CFG_{c_1}^{(x)}; \{y\}) \).
5. It follows from Proposition 1 that there exists a path from \( in \) to \( out \) in the graph \( PDG(CFG_{c_1; c_2}^{(x)}; \{y\}) \) if and only if there is such a path in the graph \( PDG(CFG_{c_1}^{(x)}; \{y\}) \) or in \( PDG(CFG_{c_2}^{(x)}; \{y\}) \) (i.e., \( x \in X_1 \cup X_2 \)), or if \( x \in \{e\} \) and there is a path from \( start \) to \( out \) in \( PDG(CFG_{c_1}^{(x)}; \{y\}) \) or in \( PDG(CFG_{c_2}^{(x)}; \{y\}) \) that contains more than 2 nodes.
6. We do a case distinction on whether there exists a path from \( start \) to \( out \) in the graph \( PDG(CFG_{c_1; c_2}^{(x)}; \{y\}) \) that contains more than 2 nodes. Assume firstly that there is such a path.
   a. Hence, by Proposition 1, there is such a path in \( PDG(CFG_{c_1}^{(x)}; \{y\}) \) or in \( PDG(CFG_{c_2}^{(x)}; \{y\}) \). Assume without loss of generality that there is such a path in \( PDG(CFG_{c_1}^{(x)}; \{y\}) \).
   b. Then, by (5), \( \Gamma = \bigcup_{x \in X_1} \Gamma(x) \).
   c. Moreover, by (4), \( \Gamma''(y) = pc \sqcup t \sqcup (\bigcup_{x \in X} \Gamma(x)) \).
(d) Moreover, by (4), either \( \Gamma'_2(y) = pc \sqcup t \sqcup (\bigcup_{x \in X_2} \Gamma(x)) \) or \( \Gamma'_2(y) = \bigcup_{x \in X_2} \Gamma(x) \).

(e) Hence, since \( \Gamma' = \Gamma'_1 \sqcup \Gamma'_2 \), it follows from (1), (c), (d), and (e) that \( \Gamma'(y) = pc \sqcup (\bigcup_{x \in X} \Gamma(x)) \).

7. Assume that there is no path \( \langle \text{start}, \ldots, \text{out} \rangle \) in PDG\( (CFG_{\Gamma_1}(z), \{y\}) \) that contains more than 2 nodes.

(a) Hence, by Proposition 1, there is no such path in PDG\( (CFG_{\Gamma_2}(z), \{y\}) \) or in PDG\( (CFG_{\Gamma_2}(z), \{y\}) \).

(b) In consequence, by (5), \( X = X_1 \cup X_2 \).

(c) Moreover, by (4), \( \Gamma'_1(y) = \bigcup_{x \in X} \Gamma(x) \) for \( i \in \{1, 2\} \).

(d) Hence, since \( \Gamma' = \Gamma'_1 \sqcup \Gamma'_2 \), it follows from (b) and (c) that \( \Gamma'(y) = \bigcup_{x \in X} \Gamma(x) \).

**Assume that** \( c = \text{while} (e) \text{ do } c_1 \text{ od} \)

1. Since \( pc \vdash \Gamma \{ \text{while} (e) \text{ do } c_1 \text{ od} \} \) \( \Gamma' \) is derivable it follows from typing rule \( \{ \text{while} \} \) that there exist \( k \in \mathbb{N} \) and sequences \( \Gamma_0, \ldots, \Gamma_{k+1}, \Gamma'_0, \ldots, \Gamma'_k \), and \( t_0, \ldots, t_k \) such that the following hold (for \( 0 \leq i \leq k \)):

   (a) \( \Gamma'_0 = \Gamma \).

   (b) \( \Gamma'_{k+1} = \Gamma'_k = \Gamma' \).

   (c) \( \Gamma'_{i+1} \vdash_{e} t_i \{ \} \).

   (d) \( pc \sqcup t_i \vdash \Gamma'_i \{ c_1 \} \Gamma''_i \), and

   (e) \( \Gamma''_{i+1} = \Gamma''_{i} \sqcup \Gamma' \).

2. We say that a loop run of the loop \( \text{while} (e) \text{ do } c_1 \text{ od} \) induces a dependency of \( z' \) on \( z \) if one of the following two conditions is satisfied:

   (a) there is a path from \( \text{in} \) to \( \text{out} \) in PDG\( (CFG_{\Gamma_1}(z), \{z'\}) \) or

   (b) \( z \in \text{fv}(e) \) and there is a path from \( \text{start} \) to \( \text{out} \) in PDG\( (CFG_{\Gamma_1}(z), \{z'\}) \) that contains more than 2 nodes.

3. Hence, by Proposition 3, \( x \in X \) if and only if there is a sequence of distinct variables \( z_1, \ldots, z_l \) such that \( x = z_1, z_l = y \), and the loop \( c \) induces a dependency of \( z_{i+1} \) on \( z_i \) for all \( i \in \{1, \ldots, l-1\} \).

4. By the induction hypothesis and (1d) it follows that, for all \( z' \in \text{Var} \) and all \( i \in \mathbb{N} \), one of the following holds, where \( Z \) is the set of all \( z \) such that there is a path from \( \text{in} \) to \( \text{out} \) in PDG\( (CFG_{\Gamma_1}(z), \{z'\}) \):

   (a) \( \Gamma''_i(z') = (\bigcup_{z' \in Z} \Gamma'_i(z')) \sqcup pc \sqcup t_i \) if there is a path from \( \text{start} \) to \( \text{out} \) in the graph PDG\( (CFG_{\Gamma_1}(z), \{z'\}) \) that contains more than 2 nodes, and

   (b) \( \Gamma''_i(z') = \bigcup_{z' \in Z} \Gamma'_i(z') \) if there is no such path.

5. From (1c) and typing rule \( \{ \text{exp} \} \) it follows that \( t_i = \bigcup_{z \in \text{fv}(e)} \Gamma'_i(z) \).

6. From the definition in (2) and from (4) and (5) it follows that for all \( z' \in \text{Var} \) one of the following holds, where \( Z' \) is the set of all \( z \) such that the loop \( c \) induces a dependency of \( z' \) on \( z \):

   (a) \( \Gamma''_i(z') = (\bigcup_{z \in Z'} \Gamma'_i(z)) \sqcup pc \) if there is a path from \( \text{start} \) to \( \text{out} \) in the graph PDG\( (CFG_{\Gamma_1}(z), \{z'\}) \) that contains more than 2 nodes, and

   (b) \( \Gamma''_i(z') = \bigcup_{z \in Z'} \Gamma'_i(z) \) if there is no such path.

7. Hence, by (1e), it follows that for all \( z \in \text{Var} \) one of the following holds:
(a) $\Gamma'_{i+1}(z') = (\bigcup_{z \in Z'} \Gamma'_i(z')) \sqcup pc \sqcup \Gamma(z')$ if there is a path from start to out in $PDG(CFG^H_{c_i}, (z_1'))$ that contains more than 2 nodes, and
(b) $\Gamma'_{i+1}(z') = (\bigcup_{z \in Z'} \Gamma'_i(z')) \sqcup \Gamma(z')$ if there is no such path.

8. Using (7) and starting with $\Gamma'(y) = \Gamma'_k(y)$, we unfold the equation further and further, thereby collecting the term $\Gamma(x)$ for all $x \in X$ on the right hand side of the equation. As a result, the following holds:
(a) $\Gamma'(y) = \bigcup_{x \in X} \Gamma(x) \sqcup pc$, if, for any $x \in \text{Var}$, there exists $z_1, \ldots, z_k \in \text{Var}$ and a path from start to out in $PDG(CFG^H_{c_i}, (z_1))$ that contains more than 2 nodes and paths from in to out in the graphs $PDG(CFG^H_{c_i}, (z_1), \ldots, (z_2))$, and
(b) $\Gamma'(y) = \bigcup_{x \in X} \Gamma(x)$ if such paths do not exist for any $x \in \text{Var}$.

9. But then, with Statement (2) of Proposition 3, it follows that
(a) $\Gamma'(y) = \bigcup_{x \in X} \Gamma(x)$, if, there exists a path from start to out in the graph $PDG(CFG^H_{c_i}, (y))$ that contains more than 2 nodes, and
(b) $\Gamma'(y) = \bigcup_{x \in X} \Gamma(x)$ if such a path does not exist. This concludes the proof.

Proof (Proof of Lemma 1 from [MS13]). Lemma 1 from [MS13] follows from Proposition 4 for $pc = I$.

2 Proof of Theorem 4 from [MS13]

To prove Theorem 4 from [MS13], we generalize the definition of $PDG^H(CFG^{H,L})$ to the form $PDG^H(CFG^{I,O}, mds)$ where $I, O$ are arbitrary sets of variables and $mds : \text{Mod} \rightarrow \mathcal{P}(\text{Var})$ is a function that specifies modes before the execution of $c$.

Definition 1. For $c \in \text{Com}$ and $mds : \text{Mod} \rightarrow \mathcal{P}(\text{Var})$ we define the function $\text{modes}_{c, mds} : (N \times \text{Mod}) \rightarrow \mathcal{P}(\text{Var})$ by $x \in \text{modes}_{c, mds}(n, m)$ if and only if for all paths (start, $n$) in $CFG_c$ one of the following two conditions is satisfied:
- there exists $n'$ on the path such that $c[n']$ acquires $m$ for $x$, and for all nodes $n''$ following $n'$ on the path $c[n'']$ does not release $m$ for $x$, or
- $x \in mds(m)$ and for all nodes $n'$ on the path $c[n']$ does not release $m$ for $x$.

Definition 2. Let $c \in \text{Com}$. Then $PDG^H(CFG^{I,O}_c, mds) = (N, E \sqcup E')$ where $(N, E) = PDG(CFG^{I,O}_c)$ and $(n, n') \in E'$ if and only if one of the following holds:
1. $n = \text{in}$ and there exist a variable $x \in I \cap \text{use}_c(n')$, a node $n'' \in N$ with $x \notin \text{modes}_{c, mds}(n'', \text{asm-nour}),$ and a path $p$ from $n''$ to $n'$ with $x \notin \text{def}_c(n''')$ for every node $n'''$ on $p$ with $n''' \neq n''$ and $n''' \neq n'$,
2. $n' = \text{out}$ and there exist a variable $x \in O \cap \text{def}_c(n')$, a node $n'' \in N$ with $x \notin \text{modes}_{c, mds}(n'', \text{asm-norw}),$ and a path $p$ from $n$ to $n''$ such that $x \notin \text{def}_c(n''')$ for every node $n'''$ on $p$ with $n''' \neq n$ and $n''' \neq n''$, or
3. $n \in \{1, \ldots, |c|\}, c[n] \in \text{Exp}$, and $n' = \text{out}$.
To prove Theorem 4 from [MS13], we establish the following more general proposition.

**Proposition 5.** Let mds be a mode state, Λ be a partial environment that is consistent with mds, and c ∈ Com. Assume that no partial environment Λ′ exists such that ⊢ Λ {c} Λ′ is derivable in the type system from [MSS11]. Then there exist x, y ∈ Var with Λ(x) = h and dom(y) = l and a path p of the form ⟨in, . . . , n, out⟩ in PDG[1](CFG[x]c, {y}, mds) where n ∈ {1, . . . , |c|} and one of the following conditions is satisfied:

1. y ∈ defc(n), there is a node n′ with y ∈ modesc,mds(n′, asm-noread), and a definition of y at n reaches n′ in CFGc, or
2. c[n] ∈ Exp.

**Proof.** The proof is by induction on the structure of the command c.

**Assume that** c = skip. Then ⊢ Λ {c} Λ is derivable, which contradicts the assumptions of the lemma.

**Assume that** c = x := e. If x ∈ dom(Λ) then ⊢ Λ {c} Λ′ is derivable for some Λ′ with rule [assign2]. In consequence, x /∈ dom(Λ). Moreover, if dom(x) = h then ⊢ Λ {c} Λ is derivable with rule [assign1]. In consequence, dom(x) = l. Moreover, if Λ(y) = l for all y ∈ fv(e) then ⊢ Λ {c} Λ would be derivable with rule [assign1]. In consequence, there exists y ∈ fv(e) with Λ(y) = h.

Since y ∈ fv(e) it follows that y ∈ usec(1), and, hence, the pair (in, 1) is an edge in PDG(CFG[y]c, {x}). In consequence, the pair is an edge in the graph PDG[1](CFG[y]c, {x}, mds).

Since x /∈ dom(Λ), Λ is consistent with mds, and dom(x) = l, it follows that x /∈ mds(asm-noread). In consequence, x /∈ modesc,mds(stop, asm-noread). Moreover, the definition of x at Node 1 reaches stop in CFGc. Hence, (1, out) is an edge in PDG[1](CFG[y]c, {x}, mds), and Condition (1) is satisfied for this edge.

Hence, (in, 1, out) is a path in PDG[1](CFG[y]c, {x}, mds) that satisfies all required conditions.

**Assume that** c = c1 ; c2. If there exist Λ′ and Λ′ such that ⊢ Λ {c1} Λ′ and ⊢ Λ′ {c2} Λ′ are derivable then ⊢ Λ {c} Λ′ is derivable with rule [seq]. In consequence, there does not exist Λ′ and Λ′ such that ⊢ Λ {c1} Λ′ and ⊢ Λ′ {c2} Λ′ are derivable. We distinguish the two cases (1) that there is no Λ′ such that ⊢ Λ {c1} Λ′ is derivable, and (2) that if ⊢ Λ {c1} Λ′ is derivable for some Λ′ then there is no Λ′ such that ⊢ Λ′ {c2} Λ′ is derivable.

1. Assume that there is no Λ′ such that ⊢ Λ {c1} Λ′ is derivable.
   (a) By the induction hypothesis there are x, y ∈ Var with Λ(x) = h and dom(y) = l and a path p from in to out in PDG[1](CFG[x]c1, {y}, mds) such that Condition (1) or Condition (2) is satisfied for the one-but-last node of p. To determine a path from in to out in PDG[1](CFG[x]c1, {y}, mds) we distinguish between whether Condition (1) or Condition (2) is satisfied for that node.
Theorem 1.1 is derivable.

Proof Sketch. Let \( P \) be the program and let \( \Pi \) be the type system. We show that \( \Pi \) is well-formed and that \( \Pi \) is sound.

1. **Well-formedness:** We show that \( \Pi \) is well-formed by verifying the typing rules and the typing judgment.

2. **Soundness:** We show that if \( \Pi \) is well-formed, then \( \Pi \) is sound. This is done by a case analysis on the typing judgments.

The details of the proof are as follows:

- **Well-formedness:**
  - \( \Pi \) is well-formed if it satisfies the typing rules.
  - We check that the typing rules are satisfied for each typing judgment.
  - This involves checking that the typing rules are satisfied for each type and term in \( \Pi \).

- **Soundness:**
  - We show that if \( \Pi \) is well-formed, then the typing judgments are satisfied.
  - We show that if the typing judgments are satisfied, then \( \Pi \) is sound.
  - This involves a case analysis on the typing judgments.

The proof is completed by verifying the well-formedness and soundness of \( \Pi \).
one of the following conditions is satisfied: \( \Lambda(e) = h \), or there is no \( \Lambda' \) such that 
\[ \vdash \Lambda \{ \{ c \} \} \Lambda' \] \( \vdash \Lambda \{ c \} \Lambda' \) is derivable, or there is no \( \Lambda' \) such that 
\[ \vdash \Lambda \{ c_2 \} \Lambda' \] \( \vdash \Lambda \{ c_2 \} \Lambda' \) is derivable.

1. Assume that \( \Lambda(e) = h \). Then there exists \( x \in fV(e) \) such that \( \Lambda(x) = h \).

Hence, \((in, 1)\) is an edge in \( PDG(\mathit{CFG}^{(x)}_{c_1}.(y)) \). In consequence, \((in, 1)\) is an edge in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \).

Let \( y \) be a variable with \( \mathit{dom}(y) = l \). Then the pair \((1, out)\) is an edge in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \) because \( c[1] \in \mathit{Exp} \).

Hence, \((in, 1, out)\) is a path from \( in \) to \( out \) in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \), and Condition (2) is satisfied for the last edge \((1, out)\).

2. Assume that there is no \( \Lambda' \) such that 
\[ \vdash \Lambda \{ c_1 \} \Lambda' \] \( \vdash \Lambda \{ c_1 \} \Lambda' \) is derivable. Then, by the induction hypothesis for \( c_1 \), there are \( x, y \) and a path \( p_1 \) in the graph \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \) with the properties stated by the lemma. But then one obtains a path \( p \) in the graph \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \) with the properties required by the lemma by increasing each node \( n \in \{1, \ldots, |c_1| \} \) in \( p_1 \) by 1.

3. Assume that there is no \( \Lambda' \) such that 
\[ \vdash \Lambda \{ c_2 \} \Lambda' \] \( \vdash \Lambda \{ c_2 \} \Lambda' \) is derivable. The proof is as in the previous case, exploiting the induction hypothesis for \( c_2 \).

**Assume that** \( c = \mathbf{while}(e)\ do\ c_1\ od: \) Since there is no \( \Lambda' \) such that the judgment 
\[ \vdash \Lambda \{ \mathbf{while}(e)\ do\ c_1\ od \} \Lambda' \] \( \vdash \Lambda \{ \mathbf{while}(e)\ do\ c_1\ od \} \Lambda' \) is derivable, by the typing rule [\text{sub}] there is no \( \Lambda'' \) with \( \Lambda \sqsubseteq \Lambda'' \sqsubseteq \Lambda' \) such that 
\[ \vdash \Lambda'' \{ \mathbf{while}(e)\ do\ c_1\ od \} \Lambda'' \] \( \vdash \Lambda'' \{ \mathbf{while}(e)\ do\ c_1\ od \} \Lambda'' \) is derivable. I.e., by the typing rule [\text{while}], for all \( \Lambda'' \) with \( \Lambda \sqsubseteq \Lambda'' \sqsubseteq \Lambda' \) one of the following conditions is satisfied: 
\[ \Lambda''(e) = h \] \( \Lambda''(e) = h \) or 
\[ \vdash \Lambda'' \{ c_1 \} \Lambda'' \] \( \vdash \Lambda'' \{ c_1 \} \Lambda'' \) is not derivable.

1. Assume that \( \Lambda''(e) = h \). Then the proof is as in the case for conditionals, defining \( p = \langle in, 1, out \rangle \).

2. Assume that \( \vdash \Lambda'' \{ c_1 \} \Lambda'' \) is not derivable. Then, by the induction hypothesis for \( c_1 \), there are \( x, y \in \mathit{Var} \) and a path \( p_1 \) in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \) with the properties stated by the lemma. But then one obtains a path \( p \) in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds) \) with the properties required by the lemma by increasing each node \( n \in \{1, \ldots, |c_1| \} \) in \( p_1 \) by 1.

Now we prove Theorem 4 from [MS13].

**Proof.** The proof is by contradiction. Assume that there is no partial environment \( \Lambda' \) such that 
\[ \vdash \Lambda_0 \{ e \} \Lambda' \] \( \vdash \Lambda_0 \{ e \} \Lambda' \) is derivable. Then, by Proposition 5, there exist \( x, y \in \mathit{Var} \) with \( \Lambda_0(x) = h \) and \( \mathit{dom}(y) = l \), such that there is a path from \( in \) to \( out \) in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds_0) \).

By the definition of \( \Lambda_0 \) we obtain that 
\[ \mathit{dom}(x) = h \]
Hence, the path in \( PDG^1(\mathit{CFG}^{(x)}_{c_1}.(y), mds_0) \) is also a path in the PDG with multi-threaded dependencies for high inputs and low outputs \( PDG_{H,I}(\mathit{CFG}_{c_1}) \).

In consequence, \( c \) is not accepted by the PDG-based analysis for threads.

### 3 Derivation Rules for the Judgment \( \langle c, \mathit{mem} \rangle \Downarrow \mathit{mem}' \)

The derivation rules for the judgment \( \langle c, \mathit{mem} \rangle \Downarrow \mathit{mem}' \) are defined in Figure 1, where \( \langle e, \mathit{mem} \rangle \Downarrow v \) denotes that expression \( e \in \mathit{Exp} \) evaluates to value \( v \in \mathit{Val} \) in memory \( \mathit{mem} \in (\mathit{Var} \rightarrow \mathit{Val}) \).
\[
\begin{align*}
\langle \text{skip, mem} \rangle & \Downarrow \text{mem} \\
\langle e, \text{mem} \rangle & \Downarrow v \\
\langle x := e, \text{mem} \rangle & \Downarrow \text{mem}[x \mapsto v] \\
\langle c_1, \text{mem} \rangle & \Downarrow \text{mem}' \\
\langle c_2, \text{mem}' \rangle & \Downarrow \text{mem}'' \\
\langle c_1; c_2, \text{mem} \rangle & \Downarrow \text{mem}''
\end{align*}
\]

**Figure 1.** Derivation rules for the judgment \( \langle e, \text{mem} \rangle \Downarrow \text{mem}' \)

References

