# Minimal Static Single Assignment Form 

Max Wagner Denis Lohner

September 13, 2023


#### Abstract

This formalization is an extension to [3]. In their work, the authors have shown that Braun et al.'s static single assignment (SSA) construction algorithm [1] produces minimal SSA form for input programs with a reducible control flow graph (CFG). However Braun et al. also proposed an extension to their algorithm that they claim produces minimal SSA form even for irreducible CFGs. In this formalization we support that claim by giving a mechanized proof.

As the extension of Braun et al.'s algorithm aims for removing socalled redundant strongly connected components (sccs) of $\phi$ functions, we show that this suffices to guarantee minimality according to Cytron et al. [2].


## Contents

1 Minimality under Irreducible Control Flow ..... 1
1.1 Proof of Lemma 1 from Braun et al. ..... 2
1.2 Proof of Minimality ..... 7

## 1 Minimality under Irreducible Control Flow

Braun et al. [1] provide an extension to the original construction algorithm to ensure minimality according to Cytron's definition even in the case of irreducible control flow. This extension establishes the property of being redundant-scc-free, i.e. the resulting graph $G$ contains no subsets inducing a strongly connected subgraph $G^{\prime}$ via $\phi$ functions such that $G^{\prime}$ has less than two $\phi$ arguments in $G \backslash G^{\prime}$. In this section we will show that a graph with this property is Cytron-minimal.

Our formalization follows the proof sketch given in [1]. We first provide a formal proof of Lemma 1 from [1] which states that every redundant set of $\phi$ functions contains at least one redundant SCC. A redundant set of $\phi$ functions is a set $P$ of $\phi$ functions with $P \cup\{v\} \supseteq A$, where $A$ is the union over all $\phi$ functions arguments contained in $P$. I.e. $P$ references at most one SSA value $(v)$ outside $P$. A redundant SCC is a redundant set that is strongly connected according to the is-argument relation.

Next, we show that a CFG in SSA form without redundant sets of $\phi$ functions is Cytron-minimal.

Finally putting those results together, we conclude that the extension to Braun et al.'s algorithm always produces minimal SSA form.

```
theory Irreducible
    imports Formal-SSA.Minimality
begin
context CFG-SSA-Transformed
begin
```


### 1.1 Proof of Lemma 1 from Braun et al.

To preserve readability, we won't distinguish between graph nodes and the $\phi$ functions contained inside such a node.

The graph induced by the $\phi$ network contained in the vertex set $P$. Note that the edges of this graph are not necessarily a subset of the edges of the input graph.
definition induced-phi-graph g $P \equiv\left\{\left(\varphi, \varphi^{\prime}\right) . \operatorname{phiArg} g \varphi \varphi^{\prime}\right\} \cap P \times P$
For the purposes of this section, we define a "redundant set" as a nonempty set of $\phi$ functions with at most one $\phi$ argument outside itself. A redundant SCC is defined analogously. Note that since any uses of values in a redundant set can be replaced by uses of its singular argument (without modifying program semantics), the name is adequate.
definition redundant-set $g P \equiv P \neq\{ \} \wedge P \subseteq \operatorname{dom}($ phi $g) \wedge\left(\exists v^{\prime} \in\right.$ allVars $g$. $\forall \varphi \in P . \forall \varphi^{\prime}$. phiArg $\left.g \varphi \varphi^{\prime} \longrightarrow \varphi^{\prime} \in P \cup\left\{v^{\prime}\right\}\right)$
definition redundant-scc g Pscc $\equiv$ redundant-set g scc $\wedge$ is-scc (induced-phi-graph $g P) s c c$

We prove an important lemma via condensation graphs of $\phi$ networks, so the relevant definitions are introduced here.
definition condensation-nodes g $P \equiv$ scc-of (induced-phi-graph g $P$ )' $P$
definition condensation-edges $g P \equiv((\lambda(x, y)$. (scc-of (induced-phi-graph g $P) x$, scc-of $($ induced-phi-graph $g P) y)$ ) (induced-phi-graph g $P))-I d$

For a finite $P$, the condensation graph induced by $P$ is finite and acyclic.
lemma condensation-finite: finite (condensation-edges g P)
The set of edges of the condensation graph, spanning at most all $\phi$ nodes and their arguments (both of which are finite sets), is finite itself.

```
proof -
    let ?phiEdges \(=\{(a, b)\). phiArg \(g\) a \(b\}\)
    have finite ?phiEdges
    proof -
        let ?phiDomRan \(=(\operatorname{dom}(p h i g) \times \bigcup(\) set' \((\operatorname{ran}(p h i g))))\)
        from phi-finite
        have finite ?phiDomRan by (simp add: imageE phi-finite map-dom-ran-finite)
        have ?phiEdges \(\subseteq\) ?phiDomRan
            apply (rule subst \([0 f \forall a \in\) ?phiEdges. \(a \in\) ?phiDomRan])
            apply (simp-all add: subset-eq[symmetric] phiArg-def)
```

```
    by (auto simp: ran-def)
    with〈finite?phiDomRan>
    show finite?phiEdges by (rule Finite-Set.rev-finite-subset)
    qed
    hence }\f\mathrm{ . finite (f '(?phiEdges }\cap(P\timesP)))\mathrm{ by auto
    thus finite (condensation-edges g P) unfolding condensation-edges-def induced-phi-graph-def
by auto
qed
    auxiliary lemmas for acyclicity
lemma condensation-nodes-edges: (condensation-edges g P)\subseteq(condensation-nodes
g P}\times\mathrm{ condensation-nodes g P)
unfolding condensation-edges-def condensation-nodes-def induced-phi-graph-def
by auto
```

lemma condensation-edge-impl-path:
assumes $(a, b) \in($ condensation-edges $g P)$
assumes $\left(\varphi_{a} \in a\right)$
assumes $\left(\varphi_{b} \in b\right)$
shows $\left(\varphi_{a}, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$
unfolding condensation-edges-def
proof -
from $\operatorname{assms}(1)$
obtain $x y$ where $x$ - $y$-props:
$(x, y) \in($ induced-phi-graph $g P)$
$a=s c c-o f($ induced-phi-graph g $P) x$
$b=$ scc-of (induced-phi-graph $g P$ ) $y$
unfolding condensation-edges-def by auto
hence $x \in a y \in b$ by auto
All that's left is to combine these paths.
with $\operatorname{assms}$ (2) $x$ - $y$-props(2)
have $\left(\varphi_{a}, x\right) \in(\text { induced-phi-graph } g P)^{*}$ by (meson is-scc-connected scc-of-is-scc)
moreover with $\operatorname{assms(3)} x$-y-props(3) $\langle y \in b\rangle$
have $\left(y, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$ by (meson is-scc-connected scc-of-is-scc)
ultimately
show $\left(\varphi_{a}, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$ using $x$ - $y$-props $(1)$ by auto
qed
lemma path-in-condensation-impl-path:
assumes $(a, b) \in(\text { condensation-edges } g P)^{+}$
assumes $\left(\varphi_{a} \in a\right)$
assumes $\left(\varphi_{b} \in b\right)$
shows $\left(\varphi_{a}, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$
using assms
proof (induction arbitrary: $\varphi_{b}$ rule:trancl-induct)
fix $y z \varphi_{b}$
assume $(y, z) \in$ condensation-edges $g P$
hence is-scc (induced-phi-graph $g P$ ) y unfolding condensation-edges-def by auto
hence $\exists \varphi_{y} . \varphi_{y} \in y$ using scc-non-empty' by auto
then obtain $\varphi_{y}$ where $\varphi_{y}$-in- $y: \varphi_{y} \in y$ by auto
assume $\varphi_{b}$-elem: $\varphi_{b} \in z$
assume $\bigwedge \varphi_{b} . \varphi_{a} \in a \Longrightarrow \varphi_{b} \in y \Longrightarrow\left(\varphi_{a}, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$
with $\operatorname{assms}(2) \varphi_{y}-i n-y$
have $\varphi_{a}$-to- $\varphi_{y}:\left(\varphi_{a}, \varphi_{y}\right) \in(\text { induced-phi-graph } g P)^{*}$ using condensation-edge-impl-path by auto
from $\varphi_{b}$-elem $\varphi_{y}$-in-y $\langle(y, z) \in$ condensation-edges $g P\rangle$
have $\left(\varphi_{y}, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$ using condensation-edge-impl-path by auto
with $\varphi_{a}-$ to $-\varphi_{y}$
show $\left(\varphi_{a}, \varphi_{b}\right) \in(\text { induced-phi-graph } g P)^{*}$ by auto
qed (auto intro:condensation-edge-impl-path)
lemma condensation-acyclic: acyclic (condensation-edges g $P$ )
proof (rule acyclicI, rule allI, rule ccontr, simp)
fix $x$
Assume there is a cycle in the condensation graph.
assume cyclic: $(x, x) \in(\text { condensation-edges } g P)^{+}$
have nonrefl: $(x, x) \notin$ (condensation-edges $g P$ ) unfolding condensation-edges-def by auto

Then there must be a second SCC $b$ on this path.
from this cyclic
obtain $b$ where $b$-on-path: $(x, b) \in($ condensation-edges $g P)(b, x) \in($ condensation-edges $g P)^{+}$
by (meson converse-tranclE)
hence $x \in($ condensation-nodes $g P) b \in($ condensation-nodes $g P)$ using con-densation-nodes-edges by auto
hence nodes-are-scc: is-scc (induced-phi-graph $g P$ ) $x$ is-scc (induced-phi-graph $g$ P) $b$
using scc-of-is-scc unfolding induced-phi-graph-def condensation-nodes-def by auto

However, the existence of this path means all nodes in $b$ and $x$ are mutually reachable.
have $\exists \varphi_{x} . \varphi_{x} \in x \exists \varphi_{b} . \varphi_{b} \in b$ using nodes-are-scc scc-non-empty' ex-in-conv by auto
then obtain $\varphi_{x} \varphi_{b}$ where $\varphi x b$-elem: $\varphi_{x} \in x \varphi_{b} \in b$ by metis
with nodes-are-scc(1) b-on-path path-in-condensation-impl-path condensation-edge-impl-path $\varphi x$-elem (2)
have $\varphi_{b} \in x$
by - (rule is-scc-closed)
This however means $x$ and $b$ must be the same SCC, which is a contradiction to the nonreflexivity of condensation-edges.
with nodes-are-scc $\varphi x b$-elem
have $x=b$ using is-scc-unique $[$ of induced-phi-graph $g P$ ] by simp
hence $(x, x) \in($ condensation-edges $g P$ ) using b-on-path by simp
with nonrefl
show False by simp
qed
Since the condensation graph of a set is acyclic and finite, it must have a leaf.

```
lemma Ex-condensation-leaf:
assumes P\not={}
shows \existsleaf.leaf }\in\mathrm{ (condensation-nodes g P)}\wedge(\forall scc.(leaf, scc) \not\in condensa
tion-edges g P)
proof -
    from assms obtain x where x\in condensation-nodes g P unfolding condensa-
tion-nodes-def by auto
    show ?thesis
    proof (rule wfE-min)
        from condensation-finite condensation-acyclic
        show wf ((condensation-edges g P )
    next
        fix leaf
        assume leaf-node:leaf \in condensation-nodes g P
        moreover
    assume leaf-is-leaf: scc # condensation-nodes g P if (scc, leaf) }\in\mathrm{ (condensation-edges
g P) -1 for scc
            ultimately
    have leaf \in condensation-nodes g P ^( }\forall\mathrm{ scc. (leaf, scc) }\not\in\mathrm{ condensation-edges
g P) using condensation-nodes-edges by blast
            thus \existsleaf.leaf \in condensation-nodes g P ^(\forallscc. (leaf, scc) #condensa-
tion-edges g P) by blast
    qed fact
qed
```

lemma scc-in- $P$ :
assumes scc $\in$ condensation-nodes $g P$
shows $s c c \subseteq P$
proof -
have $s c c \subseteq P$ if $y$-props: scc $=s c c$-of (induced-phi-graph $g P$ ) $n n \in P$ for $n$
proof -
from $y$-props
show scc $\subseteq P$
proof (clarsimp simp:y-props(1); case-tac $n=x$ )
fix $x$
assume different: $n \neq x$
assume $x \in$ scc-of (induced-phi-graph $g P$ ) $n$
hence $(n, x) \in(\text { induced-phi-graph } g P)^{*}$ by (metis is-scc-connected scc-of-is-scc node-in-scc-of-node)
with different
have $(n, x) \in(\text { induced-phi-graph } g P)^{+}$by (metis rtranclD)
then obtain $z$ where step: $(z, x) \in$ (induced-phi-graph $g P$ ) by (meson tranclE)
from step
show $x \in P$ unfolding induced-phi-graph-def by auto
qed $\operatorname{simp}$
qed
from this assms(1) have $x \in P$ if $x$-node: $x \in \operatorname{scc}$ for $x$ apply -
apply (rule imageE[of scc scc-of (induced-phi-graph g P)])
using condensation-nodes-def $x$-node by blast+
thus ?thesis by clarify
qed
lemma redundant-scc-phis:
assumes redundant-set $g$ Pscc condensation-nodes $g P x \in s c c$
shows phi $g x \neq$ None
using assms by (meson domIff redundant-set-def scc-in-P subsetCE)
The following lemma will be important for the main proof of this section. If $P$ is redundant, a leaf in the condensation graph induced by P corresponds to a strongly connected set with at most one argument, thus a redundant strongly connected set exists.

Lemma 1. Every redundant set contains a redundant SCC.

```
lemma 1:
assumes redundant-set g P
shows \existsscc\subseteqP. redundant-scc g P scc
proof -
    from assms Ex-condensation-leaf[of P g]
    obtain leaf where leaf-props:leaf }\in(\mathrm{ condensation-nodes g P) }\forall\mathrm{ scc. (leaf, scc)
 condensation-edges g P
    unfolding redundant-set-def by auto
    hence is-scc (induced-phi-graph g P) leaf unfolding condensation-nodes-def by
auto
    moreover
    hence leaf }\not={}\mathrm{ by (rule scc-non-empty')
    moreover
    have leaf \subseteqdom (phi g)
        apply (subst subset-eq, rule ballI)
        using redundant-scc-phis leaf-props(1) assms(1) by auto
    moreover
    from assms
    obtain pred where pred-props: pred \in allVars g }\forall\varphi\inP.\forall\mp@subsup{\varphi}{}{\prime}.phiArg g \varphi \varphi' \longrightarrow
\varphi
    {
```

Any argument of a $\phi$ function in the leaf SCC which is not in the leaf SCC itself must be the unique argument of P

```
    fix }\varphi\mp@subsup{\varphi}{}{\prime
    consider (in-P) \varphi' }\not=\mathrm{ leaf }\wedge\mp@subsup{\varphi}{}{\prime}\inP|(\mathrm{ neither) }\mp@subsup{\varphi}{}{\prime}\not\inleaf \wedge \varphi' & P \cup{pred}
\varphi
    hence }\mp@subsup{\varphi}{}{\prime}\inleaf\cup{\mathrm{ pred } if }\varphi\inleaf and phiArg g \varphi \varphi'
    proof cases
        case in-P - In this case leaf wasn't really a leaf, a contradiction
        moreover
        from in-P that leaf-props(1) scc-in-P[of leaf g P]
        have ( }\varphi,\mp@subsup{\varphi}{}{\prime})\in\mathrm{ induced-phi-graph g P unfolding induced-phi-graph-def by
auto
    ultimately
            have (leaf, scc-of (induced-phi-graph g P) \varphi') \in condensation-edges g P
unfolding condensation-edges-def
            using leaf-props(1) that 〈is-scc (induced-phi-graph g P) leaf〉
            apply -
            apply clarsimp
            apply (rule conjI)
            prefer 2
            apply auto[1]
            unfolding condensation-nodes-def
            by (metis (no-types, lifting) is-scc-unique node-in-scc-of-node pair-imageI
scc-of-is-scc)
            with leaf-props(2)
            show ?thesis by auto
        next
            case neither - In which case P itself wasn't redundant, a contradiction
            with that leaf-props pred-props
            have \negredundant-set g P unfolding redundant-set-def
            by (meson rev-subsetD scc-in-P)
            with assms
            show ?thesis by auto
    qed auto - the other cases are trivial
    }
    with pred-props(1)
    have \exists\mp@subsup{v}{}{\prime}\in\mathrm{ allVars g. }\forall\varphi\inleaf. \forall\mp@subsup{\varphi}{}{\prime}.\operatorname{phiArg g }\varphi\mp@subsup{\varphi}{}{\prime}\longrightarrow\mp@subsup{\varphi}{}{\prime}\inleaf \cup{\mp@subsup{v}{}{\prime}}\mathrm{ by auto}
    ultimately
    have redundant-scc g P leaf unfolding redundant-scc-def redundant-set-def by
auto
    thus ?thesis using leaf-props(1) scc-in-P by meson
qed
```


### 1.2 Proof of Minimality

We inductively define the reachable-set of a $\phi$ function as all $\phi$ functions reachable from a given node via an unbroken chain of $\phi$ argument edges to unnecessary $\phi$ functions.

```
inductive-set reachable \(::\) ' \(g \Rightarrow\) 'val \(\Rightarrow\) 'val set
```

    for \(g:: ' g\) and \(\varphi\) :: 'val
    where refl: unnecessaryPhi \(g \varphi \Longrightarrow \varphi \in\) reachable \(g \varphi\)
    \(\mid\) step: \(\varphi^{\prime} \in\) reachable \(g \varphi \Longrightarrow\) phiArg \(g \varphi^{\prime} \varphi^{\prime \prime} \Longrightarrow\) unnecessaryPhig \(\varphi^{\prime \prime} \Longrightarrow \varphi^{\prime \prime}\)
    $\in$ reachable $g \varphi$
lemma reachable-props:
assumes $\varphi^{\prime} \in$ reachable $g \varphi$
shows (phiArg g)** $\varphi \varphi^{\prime}$ and unnecessaryPhi $g \varphi^{\prime}$
using assms
by (induction $\varphi^{\prime}$ rule: reachable.induct) auto
We call the transitive arguments of a $\phi$ function not in its reachable-set the "true arguments" of this $\phi$ function.
definition [simp]: trueArgs $g \varphi \equiv\left\{\varphi^{\prime} . \varphi^{\prime} \notin\right.$ reachable $\left.g \varphi\right\} \cap\left\{\varphi^{\prime} . \exists \varphi^{\prime \prime} \in\right.$ reachable $\left.g \varphi \cdot \operatorname{phiArg} g \varphi^{\prime \prime} \varphi^{\prime}\right\}$

## lemma preds-finite: finite (trueArgs $g \varphi$ )

proof (rule ccontr)
assume infinite (trueArgs $g \varphi$ )
hence $a$ : infinite $\left\{\varphi^{\prime} . \exists \varphi^{\prime \prime} \in\right.$ reachable $g \varphi$. phiArg $\left.g \varphi^{\prime \prime} \varphi^{\prime}\right\}$ by auto
have phiarg-set: $\left\{\varphi^{\prime} . \exists \varphi\right.$. phiArg $\left.g \varphi \varphi^{\prime}\right\}=\bigcup($ set ' $\{b$. $\exists$ a. phi $g a=$ Some b $\}$ )
unfolding phiArg-def by auto
If the true arguments of a $\phi$ function are infinite in number, there must be an infinite number of $\phi$ functions...

```
    have infinite \(\left\{\varphi^{\prime} . \exists \varphi\right.\). phiArg \(\left.g \varphi \varphi^{\prime}\right\}\)
    by (rule infinite-super \(\left[\right.\) of \(\left\{\varphi^{\prime} . \exists \varphi^{\prime \prime} \in\right.\) reachable \(\left.\left.g \varphi \cdot \operatorname{phiArg} g \varphi^{\prime \prime} \varphi^{\prime}\right\}\right]\) ) (auto
simp: a)
    with phiarg-set
    have infinite (ran (phi g)) unfolding ran-def phiArg-def by clarsimp
    Which cannot be.
    thus False by (simp add:phi-finite map-dom-ran-finite)
qed
```

Any unnecessary $\phi$ with less than 2 true arguments induces with reachable $g \varphi$ a redundant set itself.
lemma few-preds-redundant:
assumes card (trueArgs $g \varphi$ ) $<2$ unnecessaryPhig $\varphi$
shows redundant-set $g$ (reachable $g \varphi$ )
unfolding redundant-set-def
proof (intro conjI)
from assms
show reachable $g \varphi \neq\{ \}$
using empty-iff reachable intros(1) by auto
next
from assms(2)

```
    show reachable g \varphi\subseteq dom (phi g)
    by (metis domIff reachable.cases subsetI unnecessaryPhi-def)
next
    from assms(1)
    consider (single) card (trueArgs g \varphi)=1|(empty)card (trueArgs g \varphi)=0 by
force
    thus \exists}\mathrm{ pred }\in\mathrm{ allVars g. }\forall\mp@subsup{\varphi}{}{\prime}\in\mathrm{ reachable g }\varphi.\forall\mp@subsup{\varphi}{}{\prime\prime}.\operatorname{phiArg g }\mp@subsup{\varphi}{}{\prime}\mp@subsup{\varphi}{}{\prime\prime}\longrightarrow\mp@subsup{\varphi}{}{\prime\prime}\in\mathrm{ reach-
able g \varphi \cup{pred}
    proof cases
        case single
    then obtain pred where pred-prop: trueArgs g \varphi ={pred} using card-eq-1-singleton
by blast
    hence pred \in allVars g by (auto intro: Int-Collect phiArg-in-allVars)
    moreover
    from pred-prop
    have }\forall\mp@subsup{\varphi}{}{\prime}\in\mathrm{ reachable g }\varphi.\forall\mp@subsup{\varphi}{}{\prime\prime}.\mathrm{ phiArg g }\mp@subsup{\varphi}{}{\prime}\mp@subsup{\varphi}{}{\prime\prime}\longrightarrow\mp@subsup{\varphi}{}{\prime\prime}\in\mathrm{ reachable g }\varphi\cup{\mathrm{ pred}
by auto
    ultimately
    show ?thesis by auto
    next
    case empty
    from allDefs-in-allVars[of-g defNode g \varphi] assms
    have phi-var: }\varphi\in\mathrm{ allVars g unfolding unnecessaryPhi-def phiDefs-def allDefs-def
defNode-def phi-def trueArgs-def
            by (clarsimp simp: domIff phis-in-\alphan)
    from empty assms(1)
    have no-preds: trueArgs g \varphi ={} by (subst card-0-eq[OF preds-finite, sym-
metric]) auto
    show ?thesis
    proof (rule bexI, rule ballI, rule allI, rule impI)
            fix }\mp@subsup{\varphi}{}{\prime}\mp@subsup{\varphi}{}{\prime\prime
            assume phis-props: }\mp@subsup{\varphi}{}{\prime}\in\mathrm{ reachable g e phiArg g }\mp@subsup{\varphi}{}{\prime}\mp@subsup{\varphi}{}{\prime\prime
            with no-preds
            have }\mp@subsup{\varphi}{}{\prime\prime}\in\mathrm{ reachable g }
            unfolding trueArgs-def
            proof -
            from phis-props
            have }\mp@subsup{\varphi}{}{\prime\prime}\in{\mp@subsup{\varphi}{}{\prime}.\exists\mp@subsup{\varphi}{}{\prime\prime}\in\mathrm{ reachable g }\varphi\mathrm{ . phiArg g }\mp@subsup{\varphi}{}{\prime\prime}\mp@subsup{\varphi}{}{\prime}}\mathrm{ by auto
            with phis-props no-preds
            show }\mp@subsup{\varphi}{}{\prime\prime}\in\mathrm{ reachable g }\varphi\mathrm{ unfolding trueArgs-def by auto
            qed
            thus }\mp@subsup{\varphi}{}{\prime\prime}\in\mathrm{ reachable g }\varphi\cup{\varphi}\mathrm{ by simp
        qed (auto simp: phi-var)
    qed
qed
lemma phiArg-trancl-same-var:
assumes (phiArg g)++}\varphi
```

shows $\operatorname{var} g \varphi=\operatorname{var} g n$
using assms
apply (induction rule: tranclp-induct)
apply (rule phiArg-same-var[symmetric])
apply simp
using phiArg-same-var by auto
The following path extension lemma will be used a number of times in the inner induction of the main proof. Basically, the idea is to extend a path ending in a $\phi$ argument to the corresponding $\phi$ function while preserving disjointness to a second path.
lemma phiArg-disjoint-paths-extend:
assumes var $g r=V$ and var $g s=V$ and $r \in$ allVars $g$ and $s \in$ allVars $g$
and $V \in$ oldDefs $g n$ and $V \in$ oldDefs $g m$
and $g \vdash n-n s \rightarrow$ defNode $g r$ and $g \vdash m-m s \rightarrow$ defNode $g s$
and set $n s \cap$ set $m s=\{ \}$
and phiArg $g \varphi_{r} r$
obtains $n s^{\prime}$
where $g \vdash n-n s @ n s^{\prime} \rightarrow$ defNode $g \varphi_{r}$
and set $\left(\right.$ butlast $\left.\left(n s @ n s^{\prime}\right)\right) \cap$ set $m s=\{ \}$
proof (cases $r=\varphi_{r}$ )
case (True)
If the node to extend the path to is already the endpoint, the lemma is trivial.
with $\operatorname{assms}(7,8,9)$ in-set-butlastD
have $g \vdash n-n s @[] \rightarrow$ defNode $g \varphi_{r}$ set (butlast $\left.(n s @[])\right) \cap$ set $m s=\{ \}$
by simp-all fastforce
with that show ?thesis .

## next

case False
It suffices to obtain any path from r to $\varphi_{r}$. However, since we'll need the corresponding predecessor of $\varphi_{r}$ later, we must do this as follows:

```
from \(\operatorname{assms}(10)\)
have \(\varphi_{r} \in\) allVars \(g\) unfolding phiArg-def
    by (metis allDefs-in-allVars phiDefs-in-allDefs phi-def phi-phiDefs phis-in- \(\alpha n\) )
    with assms(10)
    obtain \(r s^{\prime}\) pred \(_{\varphi}\) where \(r s^{\prime}\)-props: \(g \vdash\) defNode \(g r-r s^{\prime} \rightarrow\) pred \(_{\varphi_{r}}\) old.EntryPath
\(g r s^{\prime} r \in\) phiUses \(g\) pred \(_{\varphi_{r}}\) pred \(_{\varphi_{r}} \in\) set (old.predecessors \(g\left(\right.\) defNode \(\left.g \varphi_{r}\right)\) )
    by (rule phiArg-path-ex')
    define \(r s\) where \(r s=r s^{\prime} @\left[\right.\) defNode \(\left.g \varphi_{r}\right]\)
    from \(r^{\prime}\) '-props \((2,1)\) old.EntryPath-distinct old.path2-hd
    have \(r s^{\prime}\)-loopfree: defNode \(g r \notin\) set ( \(t l\) rs') by (simp add: Misc.distinct-hd-tl)
    from False assms have defNode \(g \varphi_{r} \neq\) defNode \(g r\)
    apply -
    apply (rule phiArg-distinct-nodes)
        apply (auto intro:phiArg-in-allVars)[2]
```

unfolding phiArg-def by (metis allDefs-in-allVars phiDefs-in-allDefs phi-def phi-phiDefs phis-in- $\alpha n$ )

## from $r s^{\prime}$-props

have rs-props: $g \vdash$ defNode $g r-r s \rightarrow$ defNode $g \varphi_{r}$ length $r s>1$ defNode $g r \notin$ set ( $t l r s$ )
apply (subgoal-tac defNode grehdrs')
prefer 2 using $r s^{\prime}$-props (1)
apply (rule old.path2-hd)
using old.path2-snoc old.path2-def rs'-props(1) rs-def rs'-loopfree 〈defNode g $\varphi_{r} \neq$ defNode $g r>$ by auto

```
show thesis
proof (cases set (butlast rs) \(\cap\) set \(m s=\{ \}\) )
    case inter-empty: True
```

If the intersection of these is empty, $t l r s$ is already the extension we're looking for
show thesis
proof (rule that)
show set (butlast (ns @ tl rs)) $\cap$ set $m s=\{ \}$
proof (rule ccontr, simp only: ex-in-conv[symmetric])
assume $\exists x . x \in \operatorname{set}($ butlast (ns @ tl rs)) $\cap$ set ms
then obtain $x$ where $x$-props: $x \in \operatorname{set}($ butlast ( $n s$ @ tl rs)) $x \in$ set ms by
auto with rs-props(2)
consider (in-ns) $x \in$ set $n s \mid(i n-r s) x \in \operatorname{set}$ (butlast (tl rs)) by (metis Un-iff butlast-append in-set-butlastD set-append)
thus False apply (cases)
using $x$-props(2) assms(9)
apply (simp add: disjoint-elem)
by (metis $x$-props(2) inter-empty in-set-tlD List.butlast-tl disjoint-iff-not-equal)
qed
qed (auto intro:assms(7) rs-props(1) old.path2-app)
next
case inter-ex: False
If the intersection is nonempty, there must be a first point of intersection $i$.
from inter-ex assms $(7,8)$ rs-props
obtain $i r i$ where ri-props: $g \vdash$ defNode $g r-r i \rightarrow i i \in$ set $m s \forall n \in$ set (butlast ri). $n \notin$ set ms prefix ri rs
apply -
apply (rule old.path2-split-first-prop[of $g$ defNode $g$ r rs defNode $g \varphi_{r}$, where $P=\lambda m . m \in s e t m s])$
apply blast
apply (metis disjoint-iff-not-equal in-set-butlastD)
by blast
with assms(8) old.path2-prefix-ex
obtain $m s^{\prime}$ where $m s^{\prime}$-props: $g \vdash m-m s^{\prime} \rightarrow i$ prefix $m s^{\prime} m s i \notin$ set (butlast $m s^{\prime}$ ) by blast

We proceed by case distinction:

- if $i=\operatorname{defNode} g \varphi_{r}$, the path $r i$ is already the path extension we're looking for
- Otherwise, the fact that $i$ is on the path from $\phi$ argument to the $\phi$ itself leads to a contradiction. However, we still need to distinguish the cases of whether $m=i$
consider (ri-is-valid) $i=$ defNode $g \varphi_{r} \mid\left(m\right.$-i-same) $i \neq$ defNode $g \varphi_{r} m=i$ $\mid\left(m\right.$-i-differ) $i \neq$ defNode $g \varphi_{r} m \neq i$ by auto
thus thesis
proof (cases)
case ri-is-valid
$r i$ is a valid path extension.
with $\operatorname{assms}(7)$ ri-props(1)
have $g \vdash n-n s @(t l r i) \rightarrow$ defNode $g \varphi_{r}$ by auto
moreover
have set (butlast (ns@(tl ri))) $\cap$ set $m s=\{ \}$
proof (rule ccontr)
assume contr: set (butlast (ns@ tl ri)) $\cap$ set $m s \neq\{ \}$
from this
obtain $x$ where $x$-props: $x \in \operatorname{set}($ butlast (ns @ tl ri)) $x \in$ set ms by auto with $\operatorname{assms}(9)$ have $x \notin$ set $n s$ by auto
with $x$-props $\left\langle g \vdash n-n s @ t l r i \rightarrow\right.$ defNode $\left.g \varphi_{r}\right\rangle\left\langle d e f N o d e ~ g \varphi_{r} \neq\right.$ defNode $g$
$r\rangle \operatorname{assms}(7)$
have $x \in$ set (butlast ( $t l$ ri))
by (metis Un-iff append-Nil2 butlast-append old.path2-last set-append)
with $x$-props(2) ri-props(3)
show False by (metis FormalSSA-Misc.in-set-tlD List.butlast-tl)
qed
ultimately
show thesis by (rule that)
next
case $m$ - $i$-same
If $m=i$, we have, with $m$, a variable definition on the path from a $\phi$ function to its argument. This constitutes a contradiction to the conventional property.
note $r s^{\prime}$-props(1) rs'-loopfree
moreover have $r \in$ allDefs $g$ (defNode $g r$ ) by (simp add: assms(3))
moreover from $r s^{\prime}$-props(3) have $r \in$ allUses $g$ pred $\varphi_{r}$ unfolding allUses-def by $\operatorname{simp}$


## moreover

from rs-props(1) m-i-same rs-def ri-props $(1,2,4)\left\langle d e f N o d e ~ g \varphi_{r} \neq\right.$ defNode $g$ r) $\operatorname{assms}(7,9)$
have $m \in \operatorname{set}\left(t l r s^{\prime}\right)$
by (metis disjoint-elem hd-append in-hd-or-tl-conv in-prefix list.sel(1) old.path2-hd old.path2-last old.path2-last-in-ns prefix-snoc)

## moreover

from $\operatorname{assms}(6)$ obtain $\operatorname{def}_{m}$ where $\operatorname{def}_{m} \in$ allDefs $g m$ var $g \operatorname{def}_{m}=V$ unfolding oldDefs-def using defs-in-allDefs by blast

## ultimately

have var $g \operatorname{def}_{m} \neq$ var $g r$ by $-($ rule conventional, simp-all $)$
with $\left\langle\operatorname{var} g \operatorname{def}_{m}=V\right\rangle \operatorname{assms}(1)$
have False by simp
thus ?thesis by simp

## next

case m-i-differ
If $m \neq i, i$ constitutes a proper path convergence point.
have old.pathsConverge $g m m^{\prime} n(n s @ t l r i) i$
proof (rule old.pathsConvergeI)
show $1<$ length $m s^{\prime}$ using $m$ - $i$-differ $m s^{\prime}$-props old.path2-nontriv by blast
next
show $1<l e n g t h(n s @ t l r i)$
using ri-props old.path2-nontriv assms(9) by (metis assms(7) disjoint-elem old.path2-app old.path2-hd-in-ns)
next
show set (butlast ms') $\cap$ set $($ butlast $(n s @ t l r i))=\{ \}$
proof (rule ccontr)
assume set (butlast ms $) \cap$ set (butlast (ns @ tl ri)) $\neq\{ \}$
then obtain $i^{\prime}$ where $i^{\prime}$-props: $i^{\prime} \in$ set (butlast ms') $i^{\prime} \in$ set (butlast (ns
@ tl ri)) by auto with $m s^{\prime}$-props(2)
have $i^{\prime}$-not-in-ms: $i^{\prime} \in$ set (butlast ms) by (metis in-set-butlast-appendI prefixE)

```
            with assms(9)
            show False
            proof (cases i' }\not=\mathrm{ set ns)
                    case True
                with i'-props(2)
                have \mp@subsup{i}{}{\prime}\in\mathrm{ set (butlast (tl ri))}
                    by (metis Un-iff butlast-append in-set-butlastD set-append)
                    hence i'\in set (butlast ri) by (simp add:in-set-tlD List.butlast-tl)
                    with i'-not-in-ms ri-props(3)
                    show False by (auto dest:in-set-butlastD)
            qed (meson disjoint-elem in-set-butlastD)
        qed
    qed (auto intro: assms(7) ri-props(1) old.path2-app ms'-props(1))
```

At this intersection of paths we can find a $\phi$ function.
from this assms $(6,5)$
have necessaryPhig $V i$ by (rule necessaryPhiI)
Before we can conclude that there is indeed a $\phi$ at $i$, we have to prove a couple of technicalities. .

## moreover

from $m$-i-differ ri-props $(1,4)$ rs-def old.path2-last prefix-snoc
have ri-rs'-prefix: prefix ri rs' by fastforce
then obtain $r s^{\prime}$-rest where $r s^{\prime}$-rest-prop: $r s^{\prime}=r i @ r s^{\prime}$-rest using prefixE by auto
from old.path2-last[OF ri-props(1)] last-snoc[of - i] obtain tmp where $r i=$ tmp@ $[i]$
apply (subgoal-tac $r i \neq[]$ )
prefer 2
using ri-props(1) apply (simp add: old.path2-not-Nil)
apply (rule-tac that)
using append-butlast-last-id[symmetric] by auto
with rs'-rest-prop have rs'-rest-def:rs ${ }^{\prime}=t m p @ i \# r s^{\prime}$-rest by auto
with $r s^{\prime}-\operatorname{props}(1)$ have $g \vdash i-i \# r s^{\prime}-$ rest $\rightarrow \operatorname{pred}_{\varphi r}$
by (simp add:old.path2-split)
moreover
note $\langle v a r g r=V\rangle[s i m p]$
from $r s^{\prime}-\operatorname{props}(3)$
have $r \in$ allUses $g$ pred $_{\varphi}$ unfolding allUses-def by simp
moreover
from 〈defNode gr$\neq$ set $\left.\left(t l r s^{\prime}\right)\right\rangle r s^{\prime}-r e s t-d e f$
have defNode g r $\notin$ set rs'-rest by auto
with $\left\langle g \vdash i-i \# r s^{\prime}\right.$-rest $\rightarrow$ pred $\left._{\varphi r}\right\rangle$
have $\bigwedge x . x \in$ set rs $^{\prime}$-rest $\Longrightarrow r \notin$ allDefs $g x$
by (metis defNode-eq list.distinct(1) list.sel(3) list.set-cases old.path2-cases old.path2-in- $\alpha n$ )

## moreover

from $\operatorname{assms}(7,9)\left\langle g \vdash i-i \# r s^{\prime}\right.$-rest $\rightarrow$ pred $\left._{\varphi r}\right\rangle$ ri-props(2)
have $r \notin$ defs $g i$
by (metis defNode-eq defs-in-allDefs disjoint-elem old.path2-hd-in- $\alpha$ n old.path2-last-in-ns)
ultimately
The convergence property gives us that there is a $\phi$ in the last node fulfilling necessaryPhi on a path to a use of $r$ without a definition of $r$. Thus $i$ bears a $\phi$ function for the value of $r$.
have $\exists y$. phis $g(i, r)=$ Some $y$
by (rule convergence-prop [where $g=g$ and $n=i$ and $v=r$ and $n s=i \# r s^{\prime}$-rest, simplified])

## moreover

from $\langle g \vdash n-n s \rightarrow$ defNode $g r\rangle$ have defNode $g r \in$ set $n s$ by auto
with $\langle$ set $n s \cap$ set $m s=\{ \}\rangle\langle i \in$ set $m s\rangle$ have $i \neq$ defNode $g r$ by auto moreover
from $m s^{\prime}-\operatorname{props}(1)$ have $i \in \operatorname{set}(\alpha n g)$ by auto
moreover
have defNode $g r \in \operatorname{set}(\alpha n g)$ by (simp add: assms(3))
However, we now have two definitions of $r$ : one in $i$, and one in defNode $g$ $r$, which we know to be distinct. This is a contradiction to the allDefs-disjointproperty.
ultimately have False
using allDefs-disjoint [where $g=g$ and $n=i$ and $m=d e f N o d e ~ g r]$
unfolding allDefs-def phiDefs-def
apply clarsimp
apply (erule-tac $c=r$ in equality $C E$ )
using phi-def phis-phi by auto
thus ?thesis by simp
qed
qed
qed
lemma reachable-same-var:
assumes $\varphi^{\prime} \in$ reachable $g \varphi$
shows var $g \varphi=\operatorname{var} g \varphi^{\prime}$
using assms by (metis Nitpick.rtranclp-unfold phiArg-trancl-same-var reachable-props(1))
lemma $\varphi$-node-no-defs:
assumes unnecessaryPhig $\varphi \varphi$ allVars $g$ var $g \varphi \in$ oldDefs $g n$
shows defNode $g \varphi \neq n$
using assms simpleDefs-phiDefs-var-disjoint defNode(1) not-None-eq phi-phiDefs
unfolding unnecessaryPhi-def by auto
lemma defNode-differ-aux:
assumes $\varphi_{s} \in$ reachable $g \varphi \varphi$ allVars $g s \in$ allVars $g \varphi_{s} \neq s \operatorname{var} g \varphi=\operatorname{var} g s$ shows defNode $g \varphi_{s} \neq$ defNode $g$ s unfolding reachable-def
proof (rule ccontr)
assume $\neg$ defNode $g \varphi_{s} \neq$ defNode $g s$
hence eq: defNode $g \varphi_{s}=$ defNode $g$ s by simp
from $\operatorname{assms}(1)$
have vars-eq: var $g \varphi=\operatorname{var} g \varphi_{s}$
apply -
apply (cases $\varphi=\varphi_{s}$ )
apply simp
apply (rule phiArg-trancl-same-var)
apply (drule reachable-props)
unfolding reachable-def by (meson IntD1 mem-Collect-eq rtranclpD)

```
    have \(\varphi_{s}\)-in-allVars: \(\varphi_{s} \in\) allVars \(g\) unfolding reachable-def
    proof (cases \(\varphi=\varphi_{s}\) )
    case False
    with \(\operatorname{assms}(1)\)
    obtain \(\varphi^{\prime}\) where phiArg \(g \varphi^{\prime} \varphi_{s}\) by (metis rtranclp.cases reachable-props(1))
    thus \(\varphi_{s} \in\) allVars \(g\) by (rule phiArg-in-allVars)
    next
    case eq: True
    with assms(2)
    show \(\varphi_{s} \in\) allVars \(g\) by (subst eq[symmetric])
    qed
    from \(e q \varphi_{s}\)-in-allVars \(\operatorname{assms}(3,4)\)
    have \(\operatorname{var} g \varphi_{s} \neq \operatorname{var} g s\) by \(-(\) rule defNode-var-disjoint \()\)
    with vars-eq assms(5)
    show False by auto
qed
```

Theorem 1. A graph which does not contain any redundant set is minimal according to Cytron et al.'s definition of minimality.
theorem no-redundant-set-minimal:
assumes no-redundant-set: $\neg(\exists P$. redundant-set $g P)$
shows cytronMinimal $g$
proof (rule ccontr)
assume $\neg$ cytronMinimal $g$
Assume the graph is not Cytron-minimal. Thus there is a $\phi$ function which does not sit at the convergence point of multiple liveness intervals.
then obtain $\varphi$ where $\varphi$-props: unnecessaryPhi $g \varphi \varphi \in$ allVars $g \varphi \in$ reachable $g \varphi$
using cytronMinimal-def unnecessaryPhi-def reachable-def unnecessaryPhi-def reachable.intros by auto

We consider the reachable-set of $\varphi$. If $\varphi$ has less than two true arguments, we know it to be a redundant set, a contradiction. Otherwise, we know there to be at least two paths from different definitions leading into the reachable-set of $\varphi$.
consider (nontrivial) card (trueArgs g $\varphi$ ) $\geq$ 2 $\mid($ trivial $)$ card $($ trueArgs $g \varphi)<$ 2 using linorder-not-le by auto
thus False
proof cases
case trivial
If there are less than 2 true arguments of this set, the set is trivially redundant (see few-preds-redundant).

```
from this \varphi-props(1)
    have redundant-set g (reachable g \varphi) by (rule few-preds-redundant)
    with no-redundant-set
    show False by simp
next
    case nontrivial
```

If there are two or more necessary arguments, there must be disjoint paths from Defs to two of these $\phi$ functions.
then obtain $r s \varphi_{r} \varphi_{s}$ where assign-nodes-props:
$r \neq s \varphi_{r} \in$ reachable $g \varphi \varphi_{s} \in$ reachable $g \varphi$
$\neg$ unnecessaryPhi $g r \neg$ unnecessaryPhi $g s$
$r \in\left\{n .(\text { phiArg } g)^{* *} \varphi n\right\} s \in\left\{n .(\text { phiArg } g)^{* *} \varphi n\right\}$
phiArg $g \varphi_{r} r$ phiArg $g \varphi_{s} s$
apply $\operatorname{simp}$
apply (rule set-take-two[OF nontrivial])
apply simp
by (meson reachable.intros(2) reachable-props(1) rtranclp-tranclp-tranclp tran-clp.r-into-trancl tranclp-into-rtranclp)
moreover from assign-nodes-props
have $\varphi$-r-s-uneq: $\varphi \neq r \varphi \neq s$ using $\varphi$-props by auto
moreover
from assign-nodes-props this
have $r$-s-in-tranclp: $(\text { phiArg g) })^{++} \varphi r(\text { phiArg } g)^{++} \varphi s$
by (meson mem-Collect-eq rtranclpD) (meson assign-nodes-props(7) $\varphi$-r-s-uneq(2) mem-Collect-eq rtranclpD)
from this
obtain $V$ where $V$-props: var $g r=V$ var $g s=V$ var $g \varphi=V$ by (metis phiArg-trancl-same-var)
moreover
from $r$-s-in-tranclp
have r-s-allVars: $r \in$ allVars $g s \in$ allVars $g$ by (metis phiArg-in-allVars tranclp.cases)+
moreover
from $V$-props defNode-var-disjoint $r$-s-allVars assign-nodes-props(1)
have $r$-s-defNode-distinct: defNode $g r \neq$ defNode $g$ sy auto
ultimately
obtain $n n s m m$ where $r$-s-path-props: $V \in$ oldDefs $g n g \vdash n-n s \rightarrow$ defNode $g r V \in$ oldDefs $g m \vdash \vdash m-m s \rightarrow$ defNode $g s$
set $n s \cap$ set $m s=\{ \}$ by (auto intro: ununnecessaryPhis-disjoint-paths[of $g r$ s])

```
have \(n\)-m-distinct: \(n \neq m\)
proof (rule ccontr)
    assume \(n-m\) : \(\neg n \neq m\)
    with r-s-path-props(2) old.path2-hd-in-ns
    have \(n \in\) set \(n s\) by blast
    moreover
    from \(n\)-m r-s-path-props(4) old.path2-hd-in-ns
    have \(n \in\) set ms by blast
    ultimately
    show False using r-s-path-props(5) by auto
qed
```

These paths can be extended into paths reaching $\phi$ functions in our set.
from $V$-props r-s-allVars r-s-path-props assign-nodes-props
obtain rs where rs-props: $g \vdash n-n s @ r s \rightarrow$ defNode $g \varphi_{r}$ set (butlast ( $n s @ r s$ )) $\cap$ set $m s=\{ \}$
using phiArg-disjoint-paths-extend by blast
(In fact, we can prove that set ( $n s @ r s$ ) $\cap$ set $m s=\{ \}$, which we need for the next path extension.)

$$
\text { have defNode } g \varphi_{r} \notin \text { set } m s
$$

proof (rule ccontr)
assume $\varphi_{r}$-in-ms: $\neg$ defNode $g \varphi_{r} \notin$ set $m s$
from this r-s-path-props(4)
obtain $m s^{\prime}$ where $m s^{\prime}$-props: $g \vdash m-m s^{\prime} \rightarrow$ defNode $g \varphi_{r}$ prefix $m s^{\prime} m s$ by -(rule old.path2-prefix-ex[of $g m \mathrm{~ms}$ defNode $g$ s defNode $\left.g \varphi_{r}\right]$, auto)
have old.pathsConverge $g n(n s @ r s) m m s^{\prime}\left(\right.$ defNode $\left.g \varphi_{r}\right)$
proof (rule old.pathsConvergeI)
show set (butlast (ns@rs)) $\cap$ set (butlast ms $\left.{ }^{\prime}\right)=\{ \}$
proof (rule ccontr)
assume set (butlast (ns @ rs)) $\cap$ set (butlast $\left.m s^{\prime}\right) \neq\{ \}$
then obtain $c$ where $c$-props: $c \in \operatorname{set}$ (butlast ( $n s @ r s$ )) $c \in \operatorname{set}$ (butlast $m s^{\prime}$ ) by auto
from this(2) ms'-props(2)
have $c \in$ set ms by (simp add: in-prefix in-set-butlastD)
with $c$-props(1) rs-props(2)
show False by auto
qed
next
have $m-n-\varphi_{r}$-differ: $n \neq$ defNode $g \varphi_{r} m \neq$ defNode $g \varphi_{r}$
using assign-nodes-props(2,3,4,5) $V$-props $r$-s-path-props $\varphi_{r}$-in-ms
apply fastforce
using $V$-props(1) $\varphi_{r}$-in-ms assign-nodes-props(8) old.path2-in- $\alpha n$ phiArg-def phiArg-same-var r-s-path-props $(3,4)$ simpleDefs-phiDefs-var-disjoint
by auto
with $m s^{\prime}-\operatorname{props}(1)$
show $1<$ length $m s^{\prime}$ using old.path2-nontriv by simp
from $m-n-\varphi_{r}$-differ rs-props(1)
show $1<$ length ( $n s @ r s$ ) using old.path2-nontriv by blast
qed (auto intro: rs-props set-mono-prefix $m s^{\prime}$-props)
with $V$-props $r$-s-path-props
have necessaryPhi'g $\varphi_{r}$ unfolding necessaryPhi-def using assign-nodes-props(8) phiArg-same-var by auto
with reachable-props(2)[OF assign-nodes-props(2)]
show False unfolding unnecessaryPhi-def by simp
qed
with rs-props
have aux: set ms $\cap$ set $(n s @ r s)=\{ \}$
by (metis disjoint-iff-not-equal not-in-butlast old.path2-last)
have $\varphi_{r}-V$ : var $g \varphi_{r}=V$
using $V$-props(1) assign-nodes-props(8) phiArg-same-var by auto
have $\varphi_{r}$-allVars: $\varphi_{r} \in$ allVars $g$
by (meson phiArg-def assign-nodes-props(8) allDefs-in-allVars old.path2-tl-in- $\alpha n$ phiDefs-in-allDefs phi-phiDefs rs-props)
from $V$-props(2) $\varphi_{r}-V r$-s-allVars(2) $\varphi_{r}$-allVars r-s-path-props(3) r-s-path-props(1) r-s-path-props(4) rs-props(1) aux assign-nodes-props (9)
obtain ss where ss-props: $g \vdash m-m s @ s s \rightarrow \operatorname{defNode} g \varphi_{s}$ set (butlast (ms@ss))
$\cap \operatorname{set}($ butlast $(n s @ r s))=\{ \}$
by (rule phiArg-disjoint-paths-extend) (metis disjoint-iff-not-equal in-set-butlastD)
define $p_{m}$ where $p_{m}=m s @ s s$
define $p_{n}$ where $p_{n}=n s @ r s$
have ind-props: $g \vdash m-p_{m} \rightarrow$ defNode $g \varphi_{s} g \vdash n-p_{n} \rightarrow$ defNode $g \varphi_{r}$ set (butlast $\left.p_{m}\right) \cap \operatorname{set}\left(\right.$ butlast $\left.p_{n}\right)=\{ \}$
using rs-props(1) ss-props $p_{m}$-def $p_{n}$-def by auto
The following case will occur twice in the induction, with swapped identifiers, so we're proving it outside. Basically, if the paths $p_{m}$ and $p_{n}$ intersect, the first such intersection point must be a $\phi$ function in reachable $g \varphi$, yielding the path convergence we seek.
have path-crossing-yields-convergence:
$\exists \varphi_{z} \in$ reachable $g \varphi$. $\exists \mathrm{ns}$ ms. old.pathsConverge $g n n s m \mathrm{~ms}$ (defNode $g \varphi_{z}$ )
if $\varphi_{r} \in$ reachable $g \varphi$ and $\varphi_{s} \in$ reachable $g \varphi$ and $g \vdash n-p_{n} \rightarrow$ defNode $g \varphi_{r}$ and $g \vdash m-p_{m} \rightarrow$ defNode $g \varphi_{s}$ and set (butlast $\left.p_{m}\right) \cap$ set $\left(\right.$ butlast $\left.p_{n}\right)=$
\{\}
and set $p_{m} \cap$ set $p_{n} \neq\{ \}$
for $\varphi_{r} \varphi_{s} p_{m} p_{n}$
proof -
from that(6) split-list-first-propE
obtain $p_{m} 1 n_{z} p_{m}$ 2 where $n_{z}$-props: $n_{z} \in \operatorname{set} p_{n} p_{m}=p_{m} 1 @ n_{z} \# p_{m}$ 2 $\forall n \in \operatorname{set} p_{m} 1 . n \notin \operatorname{set} p_{n}$
by (auto intro: split-list-first-propE)
with that $(3,4)$
obtain $p_{n}{ }^{\prime}$ where $p_{n}{ }^{\prime}$-props: $g \vdash n-p_{n}{ }^{\prime} \rightarrow n_{z} g \vdash m-p_{m} 1 @\left[n_{z}\right] \rightarrow n_{z}$ prefix $p_{n}{ }^{\prime} p_{n} n_{z} \notin$ set (butlast $\left.p_{n}{ }^{\prime}\right)$
by (meson old.path2-prefix-ex old.path2-split(1))
from $V$-props(3) reachable-same-var[OF that(1)] reachable-same-var[OF that(2)]
have phis- $V$ : var $g \varphi_{r}=V$ var $g \varphi_{s}=V$ by simp-all
from reachable-props(1) that (1,2) $\varphi$-props(2) phiArg-in-allVars
have phis-allVars: $\varphi_{r} \in$ allVars $g \varphi_{s} \in$ allVars $g$ by (metis rtranclp.cases) +
Various inequalities for proving paths aren't trivial.
have $n \neq$ defNode $g \varphi_{r} m \neq$ defNode $g \varphi_{r}$
using $\varphi$-node-no-defs phis-V(1) phis-allVars(1) r-s-path-props $(1,3)$ reach-able-props(2) that(1) by blast+
from $\varphi$-node-no-defs reachable-props(2) that(2) r-s-path-props(1,3) phis-V(2) that phis-allVars
have $m \neq$ defNode $g \varphi_{s} n \neq$ defNode $g \varphi_{s}$ by blast +
With this scenario, since set (butlast $\left.p_{n}\right) \cap$ set (butlast $\left.p_{m}\right)=\{ \}$, one of the paths $p_{n}$ and $p_{m}$ must end somewhere within the other, however this means the $\phi$ function in that node must either be $\varphi$ or $\varphi_{r}$
from assms $n_{z}$-props
consider $\left(p_{n}\right.$-ends-in- $\left.p_{m}\right) n_{z}=$ defNode $g \varphi_{s} \mid\left(p_{m}\right.$-ends-in- $\left.p_{n}\right) n_{z}=$ defNode $g \varphi_{r}$
proof $\left(\right.$ cases $n_{z}=$ last $\left.p_{n}\right)$
case True
with $\left\langle g \vdash n-p_{n} \rightarrow\right.$ defNode $\left.g \varphi_{r}\right\rangle$
have $n_{z}=$ defNode $g \varphi_{r}$ using old.path2-last by auto
with that(2) show ?thesis.
next
case False
from $n_{z}-\operatorname{props}(2)$
have $n_{z} \in \operatorname{set} p_{m}$ by simp
with False $n_{z}$-props $(1)\left\langle\right.$ set (butlast $\left.p_{m}\right) \cap$ set (butlast $\left.p_{n}\right)=\{ \} 〉\langle g \vdash m$ $-p_{m} \rightarrow$ defNode $g \varphi_{s}$ >
have $n_{z}=$ defNode $g \varphi_{s}$ by (metis disjoint-elem not-in-butlast old.path2-last) with that(1) show ?thesis.
qed
thus $\exists \varphi_{z} \in$ reachable $g \varphi$. $\exists \mathrm{ns}$ ms. old.pathsConverge $g n n \mathrm{~m}$ ms (defNode $g \varphi_{z}$ )
proof (cases)
case $p_{n}$-ends-in- $p_{m}$
have old.pathsConverge $g n p_{n}{ }^{\prime} m p_{m}\left(\operatorname{defNode} g \varphi_{s}\right)$
proof (rule old.pathsConvergeI)
from $p_{n}$-ends-in- $p_{m} p_{n}{ }^{\prime}$ - props(1) show $g \vdash n-p_{n}{ }^{\prime} \rightarrow$ defNode $g \varphi_{s}$ by simp
from $\left\langle n \neq\right.$ defNode $\left.g \varphi_{s}\right\rangle p_{n}$-ends-in- $p_{m} p_{n}{ }^{\prime}$-props(1) old.path2-nontriv show $1<$ length $p_{n}{ }^{\prime}$ by auto
from that(4) show $g \vdash m-p_{m} \rightarrow$ defNode $g \varphi_{s}$.
with $\left\langle m \neq\right.$ defNode $\left.g \varphi_{s}\right\rangle$ old.path2-nontriv show $1<$ length $p_{m}$ by simp
from that $p_{n}{ }^{\prime}$-props (3) show set (butlast $\left.p_{n}{ }^{\prime}\right) \cap$ set (butlast $\left.p_{m}\right)=\{ \}$
by (meson butlast-prefix disjointI disjoint-elem in-prefix)
qed
with that (1,2,3) show ?thesis by (auto intro:reachable.intros(2))
next
case $p_{m}$-ends-in- $p_{n}$
have old.pathsConverge $g n p_{n}{ }^{\prime} m\left(p_{m} 1 @\left[n_{z}\right]\right)\left(\right.$ defNode $\left.g \varphi_{r}\right)$
proof (rule old.pathsConvergeI)
from $p_{m}$-ends-in- $p_{n} \quad p_{n}{ }^{\prime}$-props $(1,2)$ show $g \vdash n-p_{n}{ }^{\prime} \rightarrow$ defNode $g \varphi_{r} g \vdash$ $m-p_{m} 1 @\left[n_{z}\right] \rightarrow$ defNode $g \varphi_{r}$ by simp-all
with $\left\langle n \neq\right.$ defNode $\left.g \varphi_{r}\right\rangle\left\langle m \neq\right.$ defNode $\left.g \varphi_{r}\right\rangle$ show $1<$ length $p_{n}{ }^{\prime} 1<$ length ( $p_{m} 1$ @ $\left[n_{z}\right]$ )
using old.path2-nontriv[of $g m p_{m} 1$ @ [ $\left.n_{z}\right]$ old.path2-nontriv[of $\left.g n\right]$ by
simp-all
from $n_{z}$-props $p_{n}{ }^{\prime}-\operatorname{props}(3)$ show set $\left(\right.$ butlast $\left.p_{n}{ }^{\prime}\right) \cap$ set (butlast ( $p_{m} 1$ @ $\left.\left.\left[n_{z}\right]\right)\right)=\{ \}$
using butlast-snoc disjointI in-prefix in-set-butlastD by fastforce
qed
with that(1) show ?thesis by (auto intro:reachable.intros)
qed
qed
Since the reachable-set was built starting at a single $\phi$, these paths must at some point converge within reachable $g \varphi$.
from assign-nodes-props(3,2) ind-props $V$-props(3) $\varphi_{r}$-V $\varphi_{r}$-allVars
have $\exists \varphi_{z} \in$ reachable $g \varphi$. $\exists \mathrm{ns}$ ms. old.pathsConverge $g n \mathrm{~ns} \mathrm{~m} \mathrm{~ms}$ (defNode $g \varphi_{z}$ )
proof (induction arbitrary: $p_{m} p_{n}$ rule: reachable.induct)
case refl
In the induction basis, we know that $\varphi=\varphi_{s}$, and a path to $\varphi_{r}$ must be obtained - for this we need a second induction.
from refl.prems refl.hyps show ?case
proof (induction arbitrary: $p_{m} p_{n}$ rule: reachable.induct)
case refl
The first case, in which $\varphi_{r}=\varphi_{s}=\varphi$, is trivial $-\varphi$ suffices.
have old.pathsConverge $g n p_{n} m p_{m}(\operatorname{defNode} g \varphi)$
proof (rule old.pathsConvergeI)
show $1<$ length $p_{n} 1<$ length $p_{m}$
using refl V-props simpleDefs-phiDefs-var-disjoint unfolding unneces-saryPhi-def
by (metis domD domIff old.path2-hd-in- $\alpha$ n old.path2-nontriv phi-phiDefs $r$-s-path-props(1) r-s-path-props(3))+
show $g \vdash n-p_{n} \rightarrow$ defNode $g \varphi g \vdash m-p_{m} \rightarrow$ defNode $g \varphi$ set (butlast $p_{n}$ )
$\cap \operatorname{set}\left(\right.$ butlast $\left.p_{m}\right)=\{ \}$
using refl by auto
qed
with $\langle\varphi \in$ reachable $g \varphi$ show ?case by auto
next
case $\left(\right.$ step $\left.\varphi^{\prime} \varphi_{r}\right)$
In this case we have that $\varphi=\varphi_{s}$ and need to acquire a path going to $\varphi_{r}$, however with the aux. lemma we have, we still need that $p_{n}$ and $p_{m}$ are disjoint.
thus ?case
proof (cases set $p_{n} \cap$ set $p_{m}=\{ \}$ )
case paths-cross: False
with step reachable.intros
show ?thesis using path-crossing-yields-convergence[of $\varphi_{r} \varphi p_{n} p_{m}$ ] by (metis disjointI disjoint-elem)

## next

case True

If the paths are intersection-free, we can apply our path extension lemma to obtain the path needed.
step (2)
from step $(9,8,10)\langle\varphi \in$ allVars $g\rangle r$-s-path-props $(1,3)$ step $(6,5)$ True
obtain $n s$ where $g \vdash n-p_{n} @ n s \rightarrow$ defNode $g \varphi^{\prime}$ set (butlast $\left.\left(p_{n} @ n s\right)\right) \cap$ set $p_{m}=\{ \}$ by (rule phiArg-disjoint-paths-extend)
from this(2) have set (butlast $\left.p_{m}\right) \cap \operatorname{set}\left(\right.$ butlast $\left.\left(p_{n} @ n s\right)\right)=\{ \}$ using in-set-butlastD by fastforce
moreover
from phiArg-same-var step.hyps(2) step.prems(5) have var g $\varphi^{\prime}=V$ by auto
moreover
have $\varphi^{\prime} \in$ allVars $g$
by (metis $\varphi$-props(2) phiArg-in-allVars reachable.cases step.hyps(1)) ultimately
show $\exists \varphi_{z} \in$ reachable $g \varphi$. $\exists \mathrm{ns}$ ms. old.pathsConverge $g n \mathrm{~ns} m \mathrm{~ms}$ (defNode $\left.g \varphi_{z}\right)$
using step.prems(1) $\varphi$-props $V$-props $\left\langle g \vdash n-p_{n} @ n s \rightarrow \operatorname{defNode} g \varphi^{\prime}\right\rangle$ by -(rule step.IH; blast)
qed
qed
next
case $\left(\right.$ step $\left.\varphi^{\prime} \varphi_{s}\right)$
With the induction basis handled, we can finally move on to the induction proper.

## show ?thesis

proof (cases set $p_{m} \cap$ set $p_{n}=\{ \}$ ) case True
have $\varphi_{s^{-}}-V: \operatorname{var} g \varphi_{s}=V$ using $\operatorname{step}(1,2,3,9)$ reachable-same-var by (simp add: phiArg-same-var)
from step(2) have $\varphi_{s}$-allVars: $\varphi_{s} \in$ allVars $g$ by (rule phiArg-in-allVars)
obtain $p_{m}{ }^{\prime}$ where tmp: $g \vdash m-p_{m} @ p_{m}{ }^{\prime} \rightarrow$ defNode $g \varphi^{\prime}$ set (butlast $\left.\left(p_{m} @ p_{m}{ }^{\prime}\right)\right) \cap \operatorname{set}\left(\right.$ butlast $\left.p_{n}\right)=\{ \}$
by (rule phiArg-disjoint-paths-extend $\left.\left[\text { of } g \varphi_{s} V \varphi_{r} m n p_{m} p_{n} \varphi\right]_{)}\right)$
(metis $\varphi_{s}-V \varphi_{s^{-}}$allVars step $r$-s-path-props $(1,3)$ True disjoint-iff-not-equal in-set-butlastD)+
from step(5) this(1) step(7) this(2) step (9) step(10) step(11)
show ?thesis by (rule step.IH[of $\left.p_{m} @ p_{m}{ }^{\prime} p_{n}\right]$ )
next
case paths-cross: False
with step reachable.intros
show ?thesis using path-crossing-yields-convergence $\left[\right.$ of $\left.\varphi_{r} \varphi_{s} p_{n} p_{m}\right]$ by blast
qed
qed
then obtain $\varphi_{z} n s m s$ where $\varphi_{z} \in$ reachable $g \varphi$ and old.pathsConverge $g n$ $n s m$ ms (defNode $g \varphi_{z}$ )
by blast
moreover
with reachable-props have var $g \varphi_{z}=V$ by (metis V-props(3) phiArg-trancl-same-var

$$
\text { rtranclp } D)
$$

ultimately have necessaryPhi' $g \varphi_{z}$ using $r$-s-path-props
unfolding necessaryPhi-def by blast
moreover with $\left\langle\varphi_{z} \in\right.$ reachable $\left.g \varphi\right\rangle$ have unnecessaryPhi $g \varphi_{z}$ by $-($ rule reachable-props)
ultimately show False unfolding unnecessaryPhi-def by blast qed
qed
Together with lemma 1, we thus have that a CFG without redundant SCCs is cytron-minimal, proving that the property established by Braun et al.s algorithm suffices.
corollary no-redundant-SCC-minimal:
assumes $\neg(\exists P$ scc. redundant-scc g $P$ scc $)$
shows cytronMinimal $g$
using assms 1 no-redundant-set-minimal by blast
Finally, to conclude, we'll show that the above theorem is indeed a stronger assertion about a graph than the lack of trivial $\phi$ functions. Intuitively, this is because a set containing only a trivial $\phi$ function is a redundant set.

```
corollary
assumes }\neg(\existsP\mathrm{ . redundant-set g P)
shows \negredundant g
proof -
    have redundant g\Longrightarrow\existsP. redundant-set g P
    proof -
    assume redundant g
    then obtain \varphi where phig \varphi\not= None trivial g \varphi
    unfolding redundant-def redundant-set-def dom-def phiArg-def trivial-def isTriv-
ialPhi-def
            by (clarsimp split: option.splits) fastforce
            hence redundant-set g{\varphi}
            unfolding redundant-set-def dom-def phiArg-def trivial-def isTrivialPhi-def
            by auto
    thus ?thesis by auto
    qed
    with assms show ?thesis by auto
qed
end
end
```


## References

[1] M. Braun, S. Buchwald, S. Hack, R. Leißa, C. Mallon, and A. Zwinkau. Simple and efficient construction of static single assignment form. In R. Jhala and K. Bosschere, editors, Compiler Construction, volume 7791 of Lecture Notes in Computer Science, pages 102-122. Springer Berlin Heidelberg, 2013.
[2] R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman, and F. K. Zadeck. Efficiently computing static single assignment form and the control dependence graph. ACM Transactions on Programming Languages and Systems, 13(4):451-490, Oct. 1991.
[3] S. Ullrich and D. Lohner. Verified construction of static single assignment form. Archive of Formal Proofs, Feb. 2016. http://isa-afp.org/entries/ Formal_SSA.shtml, Formal proof development.

