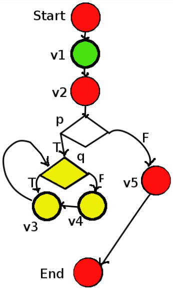
Generalising Control Dependence

10th CREST Open Workshop Program Analysis and Slicing

Sebastian Danicic

Goldsmiths, University of London

25th January 2011



Co-Authors

Richard W. Barraclough	@UK PLC,
Mark Harman	Crest, University College London, UK
Ákos Kiss	University of Szeged, Hungary,
Michael R. Laurence	University of Sheffield, UK

Control Dependence - a Brief History

Control Dependence - a Brief History

1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM

Control Dependence - a Brief History

1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM

1979 Weiser: PhD Thesis First use of Control Dependence in Slicing - although called something else.

Control Dependence - a Brief History

1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM

1979 Weiser: PhD Thesis First use of Control Dependence in Slicing - although called something else.

1987 J. Ferrante, K. J. Ottenstein, J. D. Warren, The program dependence graph and its use in optimization, TOPLAS. First to use the term Control Dependence

Control Dependence - a Brief History

1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM

1979 Weiser: PhD Thesis First use of Control Dependence in Slicing - although called something else.

1987 J. Ferrante, K. J. Ottenstein, J. D. Warren, The program dependence graph and its use in optimization, TOPLAS. First to use the term Control Dependence

1990 A. Podgurski, L. Clarke, A formal model of program dependences and its implications for software testing, debugging, and maintenance, TSE.

Control Dependence - a Brief History

- 1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM
- 1979 Weiser: PhD Thesis First use of Control Dependence in Slicing - although called something else.
- 1987 J. Ferrante, K. J. Ottenstein, J. D. Warren, The program dependence graph and its use in optimization, TOPLAS. First to use the term Control Dependence
- 1990 A. Podgurski, L. Clarke, A formal model of program dependences and its implications for software testing, debugging, and maintenance, TSE.
- 1996 G. Bilardi, K. Pingali, A framework for generalized control dependence, in: PLDI.

Control Dependence - a Brief History

1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM

1979 Weiser: PhD Thesis First use of Control Dependence in Slicing - although called something else.

1987 J. Ferrante, K. J. Ottenstein, J. D. Warren, The program dependence graph and its use in optimization, TOPLAS. First to use the term Control Dependence

1990 A. Podgurski, L. Clarke, A formal model of program dependences and its implications for software testing, debugging, and maintenance, TSE.

1996 G. Bilardi, K. Pingali, A framework for generalized control dependence, in: PLDI.

2007 V. P. Ranganath et al., A new foundation for control dependence and slicing for modern program structures, TOPLAS.

Control Dependence - a Brief History

1977 D. E. Denning, P. J. Denning,Certification of programs for secure information flow Communications of the ACM

1979 Weiser: PhD Thesis First use of Control Dependence in Slicing - although called something else.

1987 J. Ferrante, K. J. Ottenstein, J. D. Warren, The program dependence graph and its use in optimization, TOPLAS. First to use the term Control Dependence

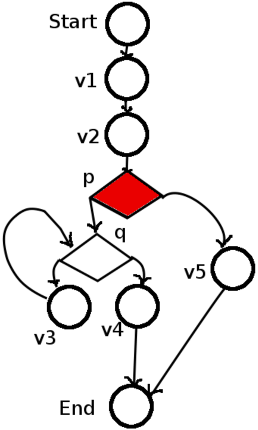
1990 A. Podgurski, L. Clarke, A formal model of program dependences and its implications for software testing, debugging, and maintenance, TSE.

1996 G. Bilardi, K. Pingali, A framework for generalized control dependence, in: PLDI.

2007 V. P. Ranganath et al., A new foundation for control dependence and slicing for modern program structures, TOPLAS.

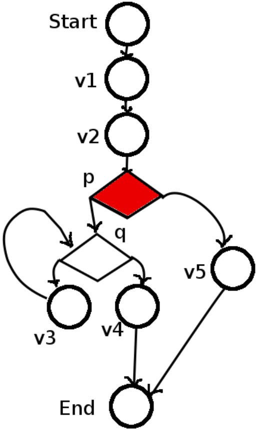
2008 T. Amtoft, Slicing for modern program structures: a theory for eliminating irrelevant loops, IPL.

Traditional (Ferrante et al.) Control Dependence



Node p controls node v if there is a path from p to **end** which does not pass through v but there is a successor of p from which all paths to **end** go through v.

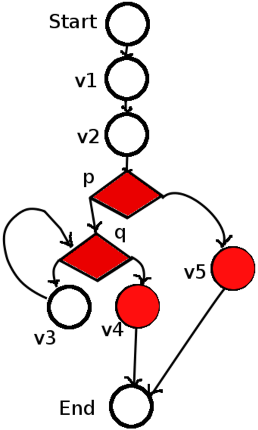
Traditional (Ferrante et al.) Control Dependence



- Which nodes does p control?

Node p controls node v if there is a path from p to **end** which does not pass through v but there is a successor of p from which all paths to **end** go through v.

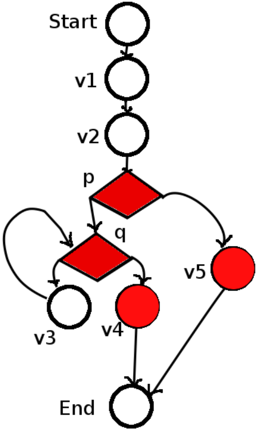
Traditional (Ferrante et al.) Control Dependence



- Which nodes does p control?
- $\{q, v_4, v_5\}$,

Node p controls node v if there is a path from p to **end** which does not pass through v but there is a successor of p from which all paths to **end** go through v.

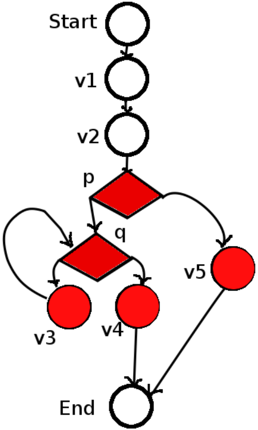
Traditional (Ferrante et al.) Control Dependence



- Which nodes does p control?
- $\{q, v_4, v_5\}$,
- but not v_3.

Node p controls node v if there is a path from p to **end** which does not pass through v but there is a successor of p from which all paths to **end** go through v.

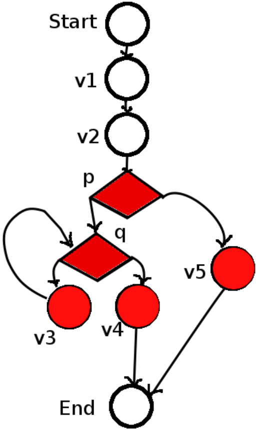
Traditional (Ferrante et al.) Control Dependence



- Which nodes does p control?
- $\{q, v_4, v_5\}$,
- but not v_3.
- q controls v_3.

Node p controls node v if there is a path from p to **end** which does not pass through v but there is a successor of p from which all paths to **end** go through v.

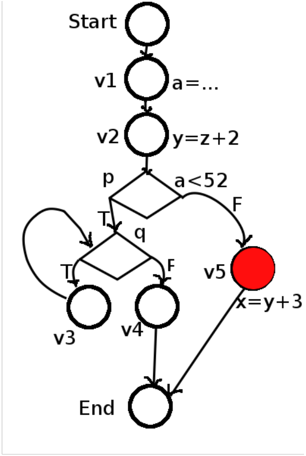
Traditional (Ferrante et al.) Control Dependence



- Which nodes does p control?
- $\{q, v_4, v_5\}$,
- but not v_3.
- q controls v_3.
- So p transitively controls v_3.

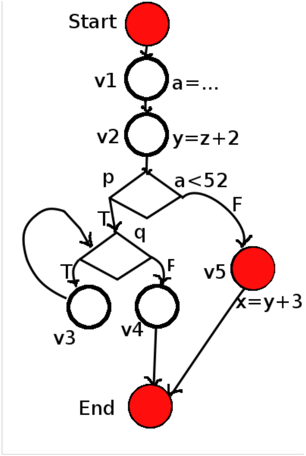
Node p controls node v if there is a path from p to **end** which does not pass through v but there is a successor of p from which all paths to **end** go through v.

Slicing - Data and Control Dependence



Slice at v_5.

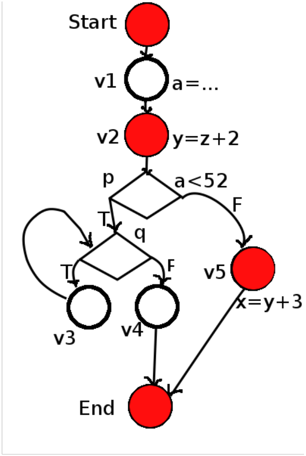
Slicing - Data and Control Dependence



Slice at v_5.

1. First add **start** and **end**.

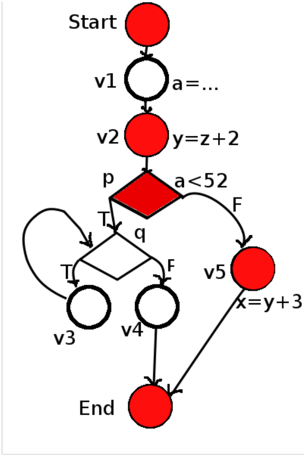
Slicing - Data and Control Dependence



Slice at v_5.

1. First add **start** and **end**.
2. v_5 is data dependent on v_2.

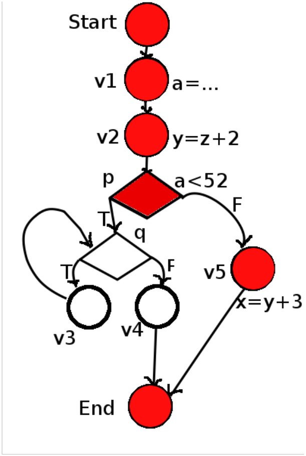
Slicing - Data and Control Dependence



Slice at v_5.

1. First add **start** and **end**.
2. v_5 is data dependent on v_2.
3. v_5 is control dependent on p.

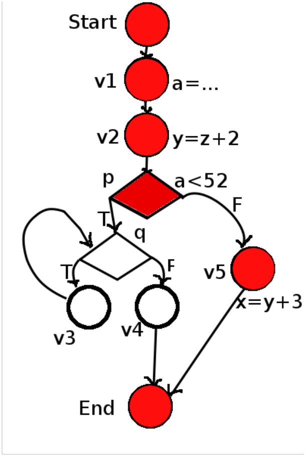
Slicing - Data and Control Dependence



Slice at v_5.

1. First add **start** and **end**.
2. v_5 is data dependent on v_2.
3. v_5 is control dependent on p.
4. p is data dependent on v_1.

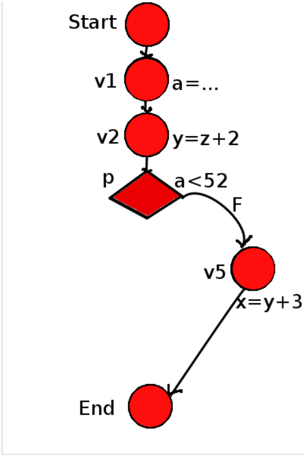
Slicing - Data and Control Dependence



Slice at v_5.

1. First add **start** and **end**.
2. v_5 is data dependent on v_2.
3. v_5 is control dependent on p.
4. p is data dependent on v_1.
5. The set of red nodes is closed under control and data dependence

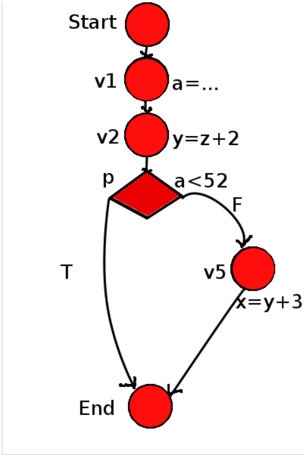
Slicing - Data and Control Dependence



Slice at v_5.

1. First add **start** and **end**.
2. v_5 is data dependent on v_2.
3. v_5 is control dependent on p.
4. p is data dependent on v_1.
5. The set of red nodes is closed under control and data dependence
6. Remove the non-red nodes.

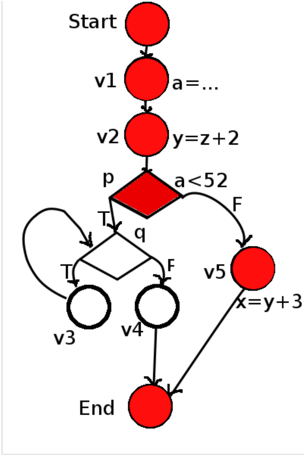
Slicing - Data and Control Dependence



Slice at v_5.

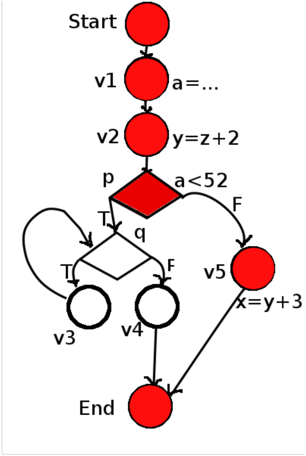
1. First add **start** and **end**.
2. v_5 is data dependent on v_2.
3. v_5 is control dependent on p.
4. p is data dependent on v_1.
5. The set of red nodes is closed under control and data dependence
6. Remove the non-red nodes.
7. Finally, 'rewire' the graph.

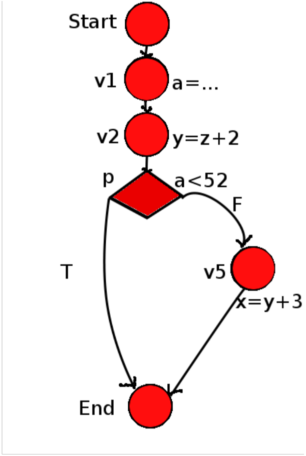
Rewiring



To rewire, add an edge between two red nodes if there is a path with no intervening red nodes. We label it with the same label as the initial edge.

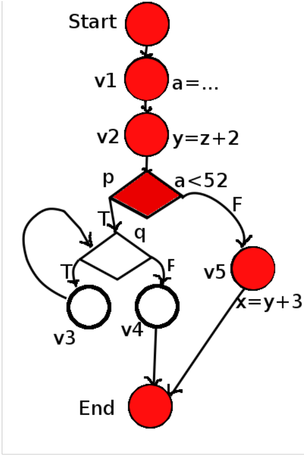
The Induced Graph

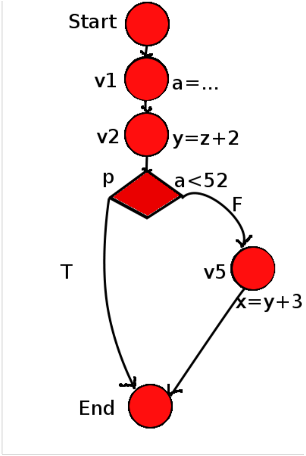




To rewire, add an edge between two red nodes if there is a path with no intervening red nodes. We label it with the same label as the initial edge.

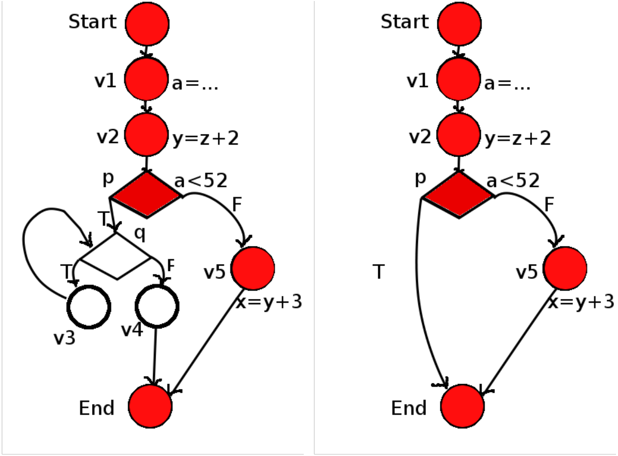
Slicing





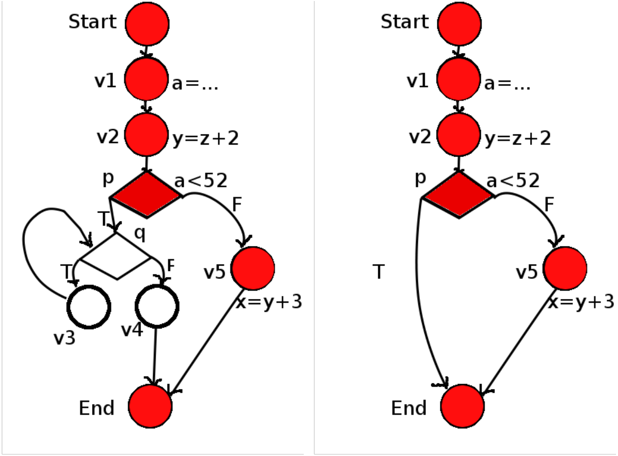
So slicing involves computing a set closed under control and data and then building the induced graph.

Slicing



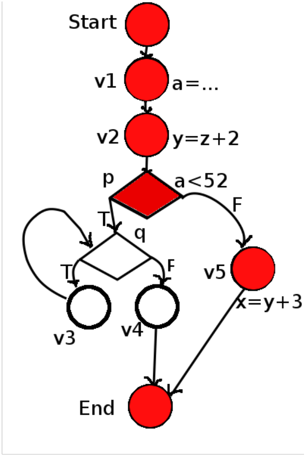
Notice: non-termination may not be preserved.

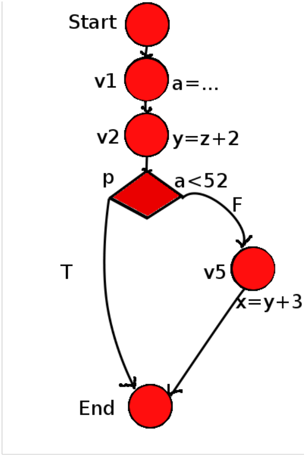
Slicing



Notice: non-termination may not be preserved. This is because traditional control dependence is 'non-termination insensitve'.

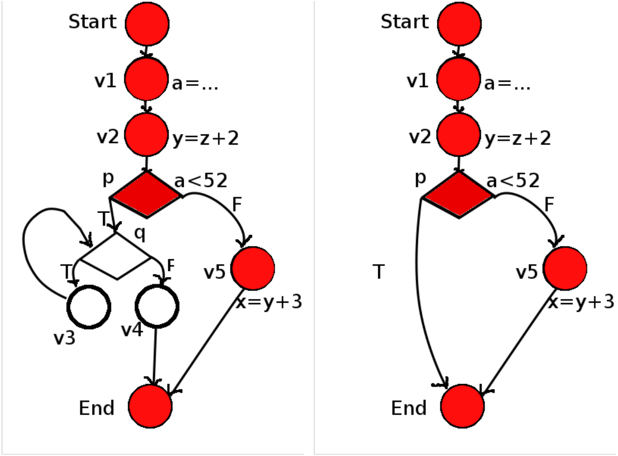
Slicing





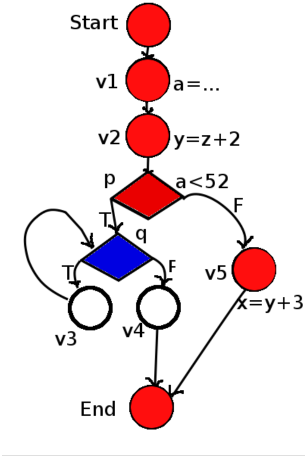
Notice: non-termination may not be preserved. This is because traditional control dependence is 'non-termination insensitve'. We prefer to call it **weak**.

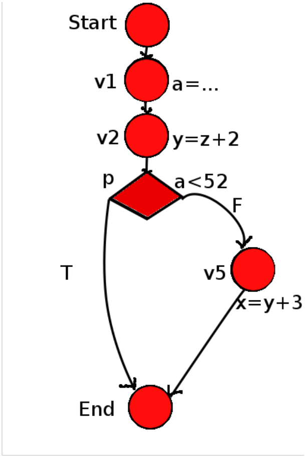
Slicing



What if you want slices to preserve non-termination?

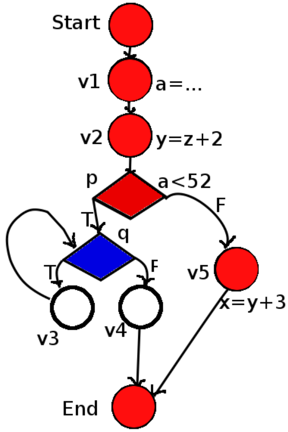
Slicing

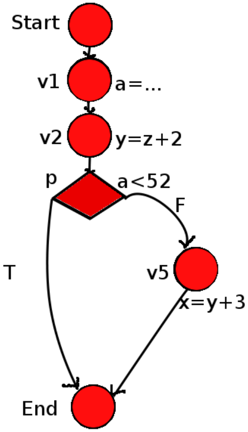




What if you want slices to preserve non-termination? We need q to be included too.

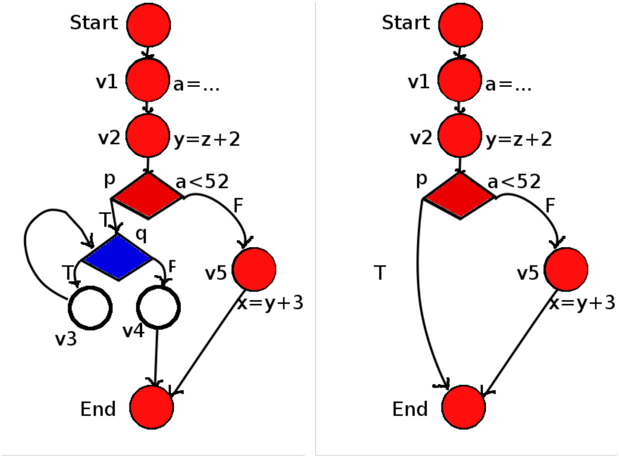
Slicing





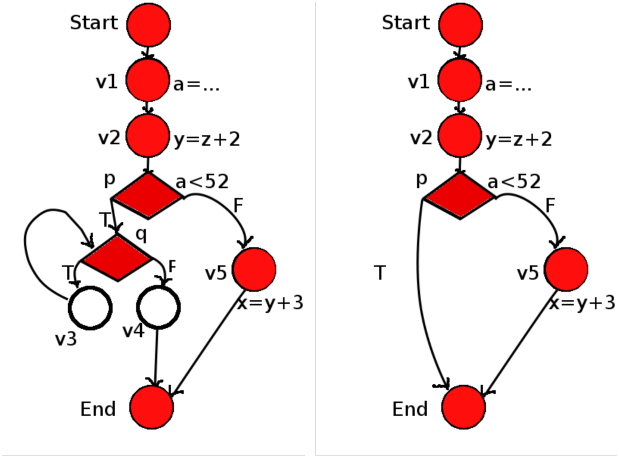
but q does not control anything using the traditional definitions of Ferrante and Ottenstein (1987) and previously Weiser (1981).

Slicing



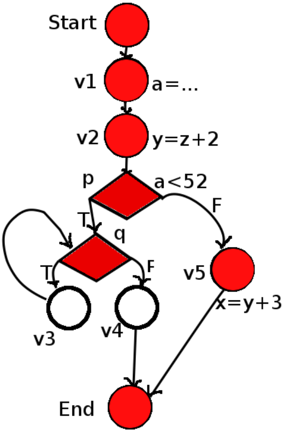
Podgurski and Clarke (1990) introduced a form of control dependence which solved this problem.

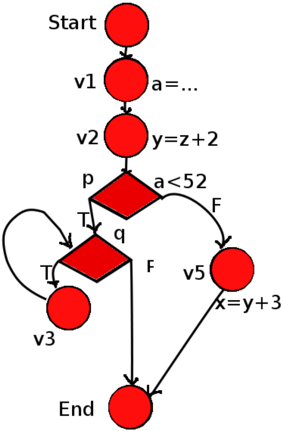
Slicing



Podgurski and Clarke (1990) introduced a form of control dependence which solved this problem. *q* controls **end** using their definition.

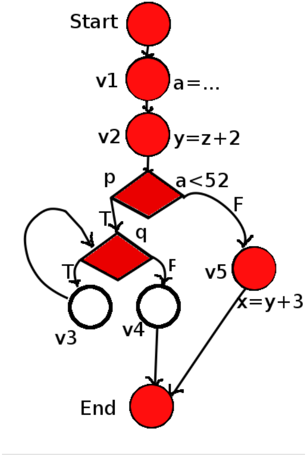
Slicing

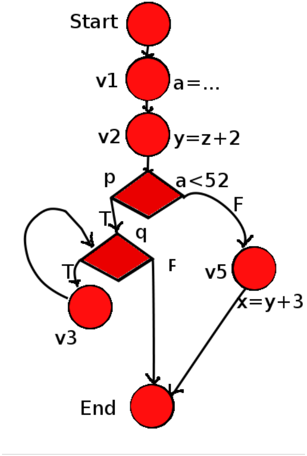




the slice produced using Podgurski and Clarke's control dependence preserves non-termination.

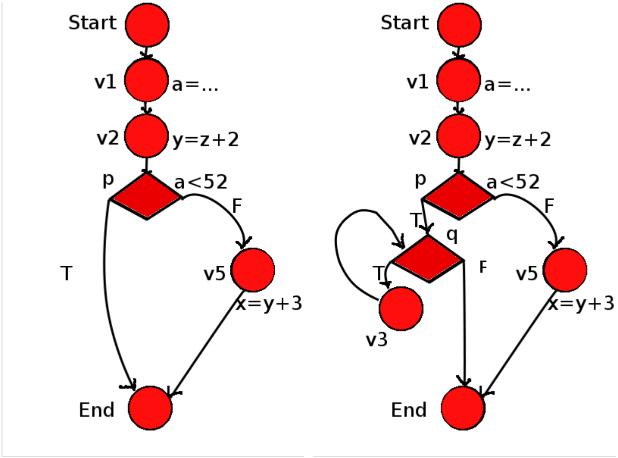
Slicing





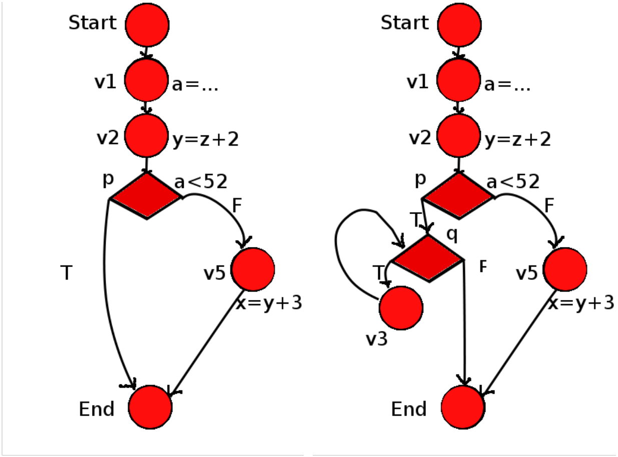
Theirs is a ‘non-termination sensitive’ or as we prefer, **strong** form of control dependence.

Two Forms of Slice



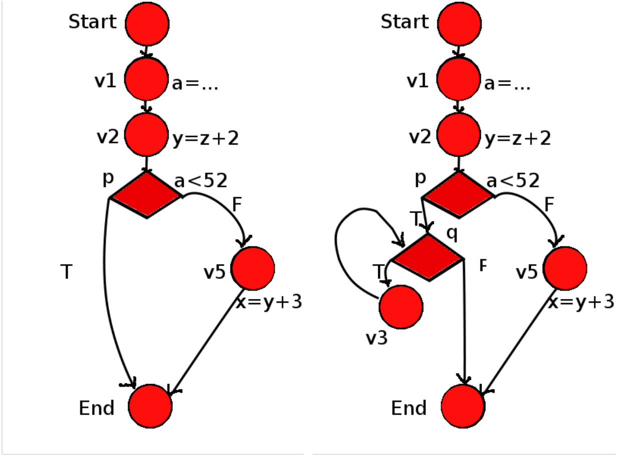
The weak slice and the strong slice

Slicing



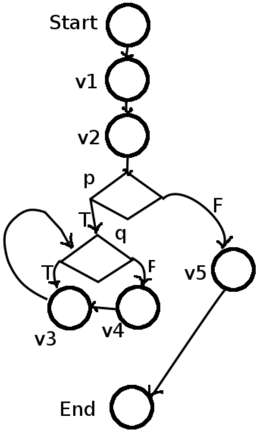
but Podgurski and Clarke's definition only works if **end** is reachable from every node.

Slicing

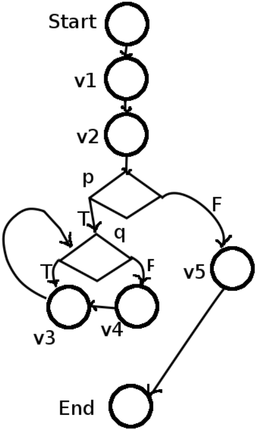


but Podgurski and Clarke's definition only works if **end** is reachable from every node. This is not the case in reactive systems.

Slicing Reactive Systems

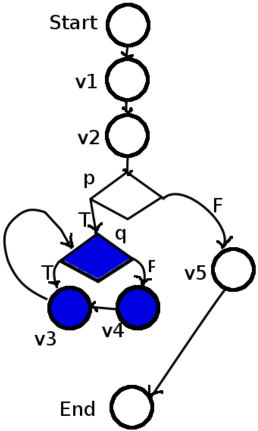


Slicing Reactive Systems



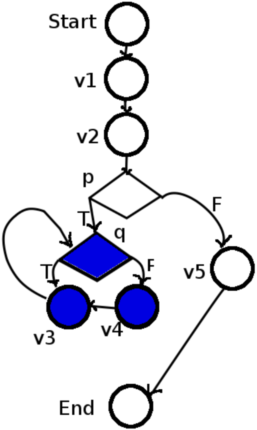
In reactive systems we have intentionally non-terminating programs.

Slicing Reactive Systems



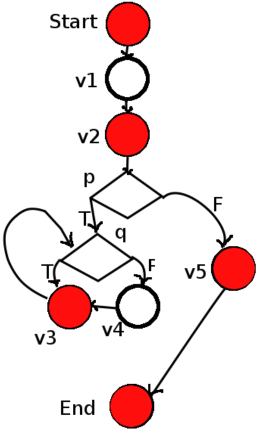
In reactive systems we have intentionally non-terminating programs. Here we have a ‘deliberate’ infinite loop.

Slicing Reactive Systems



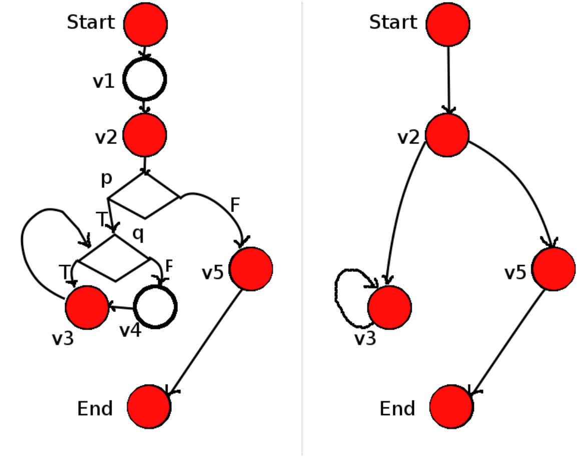
In reactive systems we have intentionally non-terminating programs. Here we have a 'deliberate' infinite loop. This is a problem.

Slicing Reactive Systems



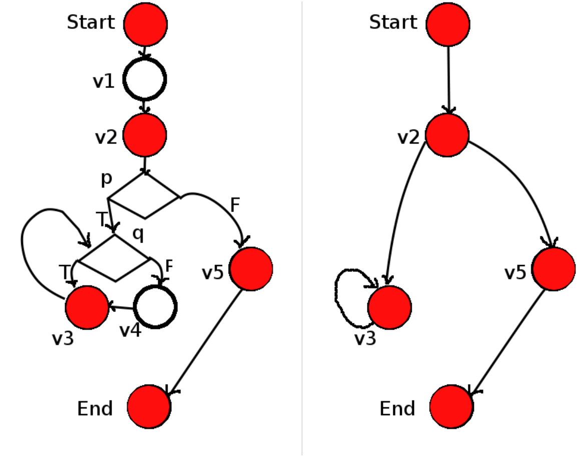
The red set is closed under traditional control dependence and also under Podgurski and Clarke's control dependence.

Slicing Reactive Systems



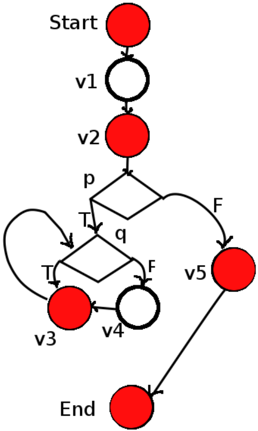
The red set is closed under traditional control dependence and also under Podgurski and Clarke's control dependence. But ...

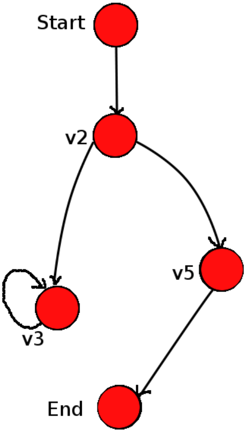
Slicing Reactive Systems



The induced graph isn't even a legal CFG. v_2 is a non-predicate of out degree greater than one.

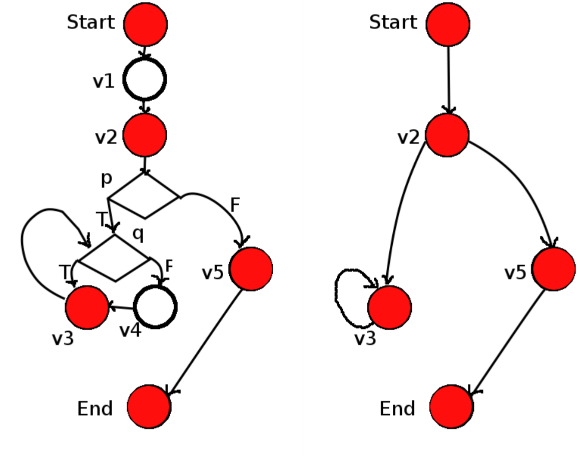
Slicing Reactive Systems





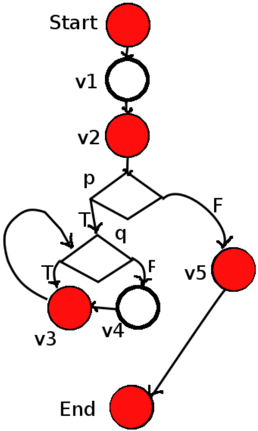
Ranganath et al. (2007) noticed that we need new forms of control dependence to solve this problem.

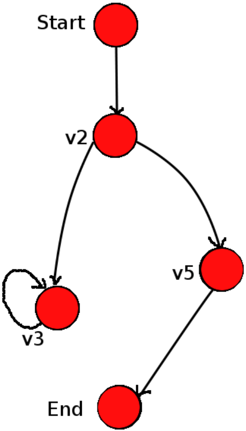
Slicing Reactive Systems



They introduced $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ which produced strong slices for reactive systems. (A generalisation of Podgurski and Clarke's definition).

Slicing Reactive Systems





Later Amtoft (2008) produced $\xrightarrow{\text{WOD}}$ which gives rise to weak slices of reactive systems. (A generalisation of Ferrante et al.'s definition).

Contributions of our Work

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

Contributions of our Work

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way?

Contributions of our Work

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way? yes!

Contributions of our Work

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way? yes!
- Are there underlying semantic properties captured by all these different forms of control dependence?

Contributions of our Work

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way? yes!
- Are there underlying semantic properties captured by all these different forms of control dependence? yes!

Categorisation of the Different Forms of Control Dependence

- Weak (Non-termination sensitive):

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

Categorisation of the Different Forms of Control Dependence

- Weak (Non-termination sensitive):

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Strong (Non-termination sensitive):

$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)

Weak Commitment-Closedness

- We do not give yet another definition of control dependence.

Weak Commitment-Closedness

- We do not give yet another definition of control dependence.
- Instead we give a property of sets closed under non-termination insensitive control dependence.

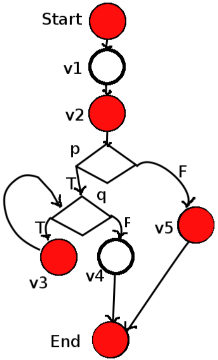
Weak Commitment-Closedness

- We do not give yet another definition of control dependence.
- Instead we give a property of sets closed under non-termination insensitive control dependence.
- The sets are **Weak commitment-closed**

Weak Commitment-Closedness

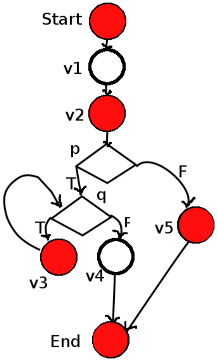
- We do not give yet another definition of control dependence.
- Instead we give a property of sets closed under non-termination insensitive control dependence.
- The sets are **Weak commitment-closed**
- This definition works for all directed graphs and is hence more general.

Definition: *S*-Weakly Committing Nodes



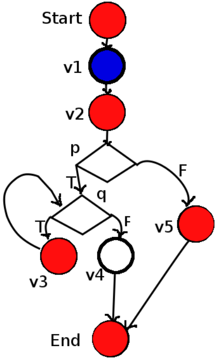
A node is *S*-**weakly committing** if on every path from it we reach the same element of *S* first.

Definition: S-Weakly Committing Nodes



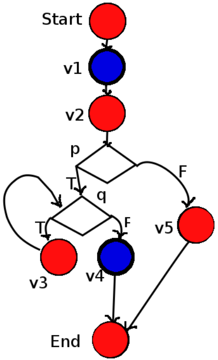
A node is S-**weakly committing** if on every path from it we reach the same element of S first. Trivially, all elements of S are S-weakly committing.

Definition: S-Weakly Committing Nodes



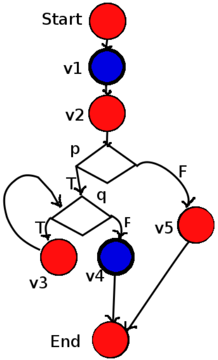
A node is S-**weakly committing** if on every path from it we reach the <ins>same</ins> element of S first. Trivially, all elements of S are S-weakly committing. v_1 is S-weakly committing, since we always reach v_2 first.

Definition: S-Weakly Committing Nodes



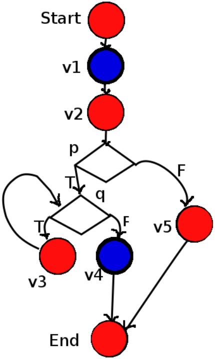
A node is S-**weakly committing** if on every path from it we reach the <u>same</u> element of S first. Trivially, all elements of S are S-weakly committing. v_1 is S-weakly committing, since we always reach v_2 first. So is v_4.

Definition: S-Weakly Committing Nodes



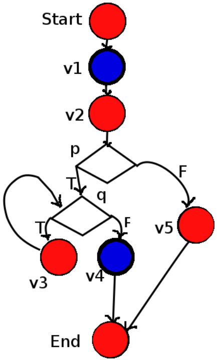
A node is S-**weakly committing** if on every path from it we reach the same element of S first. Trivially, all elements of S are S-weakly committing. v_1 is S-weakly committing, since we always reach v_2 first. So is v_4. Nodes p and q are not weakly committing.

Definition: Weakly Commitment-closed Sets



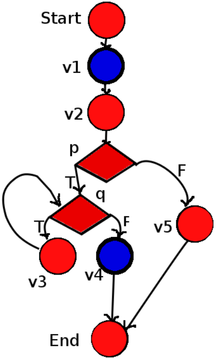
A set S is **weakly commitment-closed** if all nodes not in S are S-weakly committing.

Definition: Weakly Commitment-closed Sets



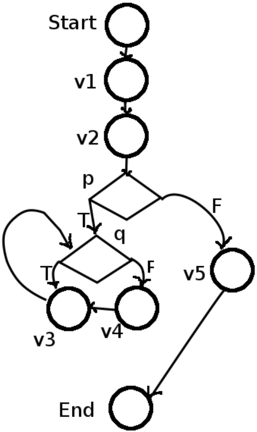
A set S is **weakly commitment-closed** if all nodes not in S are S-weakly committing. This S is not weakly commitment-closed.

Definition: Weakly Commitment-closed Sets



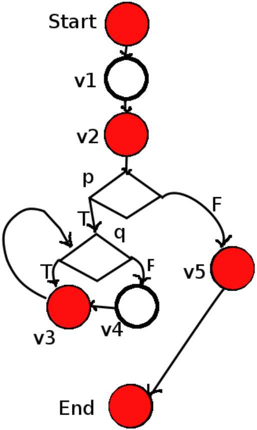
A set S is **weakly commitment-closed** if all nodes not in S are S-weakly committing. This S is not weakly commitment-closed. Now it is!

Weakly Commitment-closed Sets in Reactive Systems



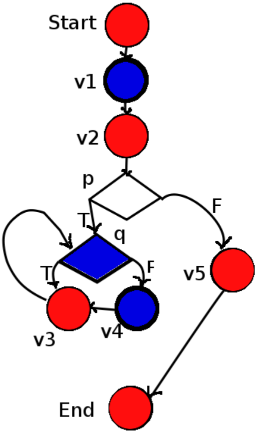
So let's see how it works for reactive systems.

Weakly Commitment-closed Sets in Reactive Systems



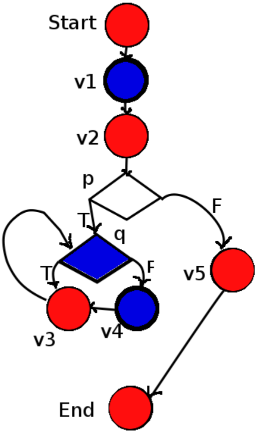
Which nodes are S-weakly committing?

Weakly Commitment-closed Sets in Reactive Systems



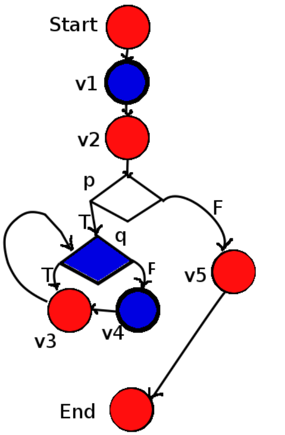
Which nodes are S-weakly committing? v_1, q and v_4.

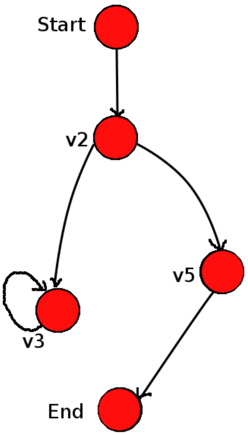
Weakly Commitment-closed Sets in Reactive Systems



Which nodes are S-weakly committing? v_1, q and v_4. But not p. So S is not weak commitment-closed.

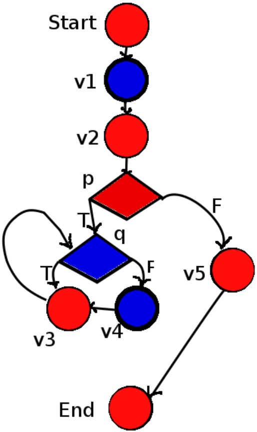
Weakly Commitment-closed Sets in Reactive Systems





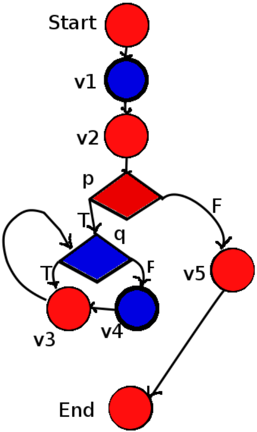
Which nodes are S-weakly committing? v_1, q and v_4. But not p. So S is not weak commitment-closed. So the induced graph is bad.

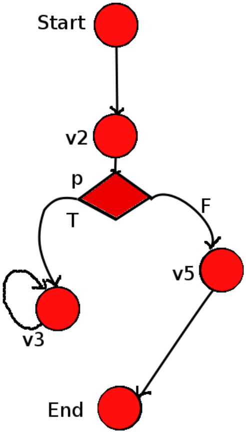
Weakly Commitment-closed Sets in Reactive Systems



Now S is weakly commitment-closed!

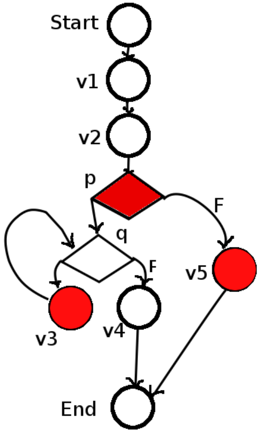
Weakly Commitment-closed Sets in Reactive Systems





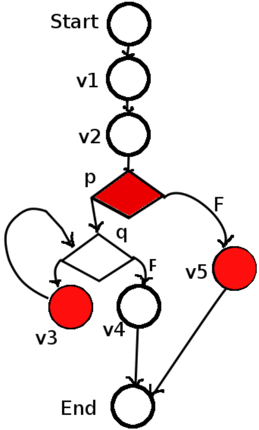
Now *S* is weak commitment-closed! So the induced graph is good.

Theorem 1: Soundness and Completeness of WCC



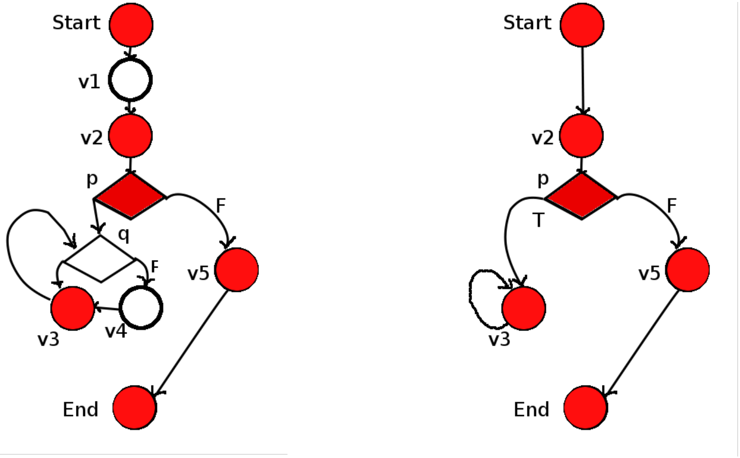
For each weak form of control dependence c in the literature, a set S is closed under c if and only if S is weakly commitment-closed.

Generality of WCC



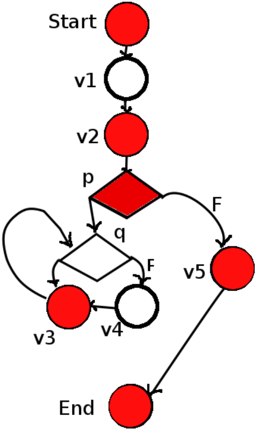
The beauty of weak commitment-closedness is that there is no need to consider special cases considered by previous authors. It works for them all.

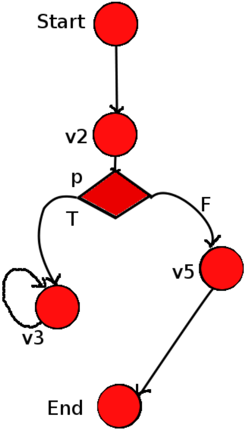
Generality of WCC



Using Weak Commitment-Closedness, things like **end** reachability are irrelevant. It ‘works’ for all directed graphs.

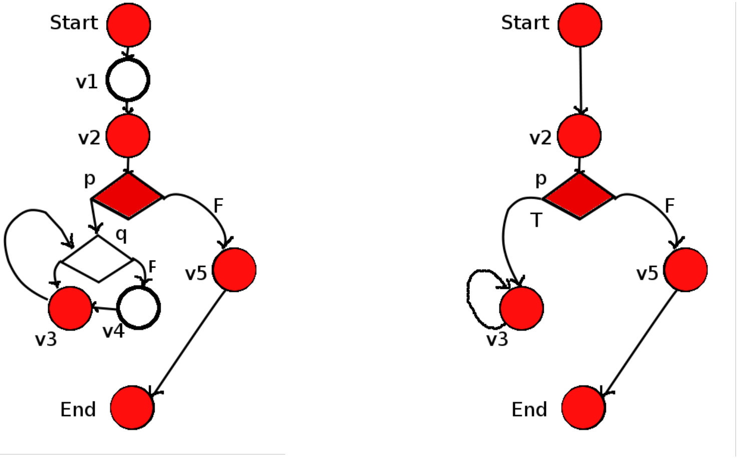
Algorithm for WCC





We have an algorithm $O(n^3)log(n)$ which given any node set V, computes the minimal weakly commitment closed set containing V.

Using WCC



Because of Theorem 1, this algorithm can be used in all cases instead of the weak forms of control dependence in the literature.

Traditional Slicing using Weakly Commitment-closed Sets

- So in traditional slicing, given a slicing criterion V' we must find the minimal weakly commitment closed set containing V'.

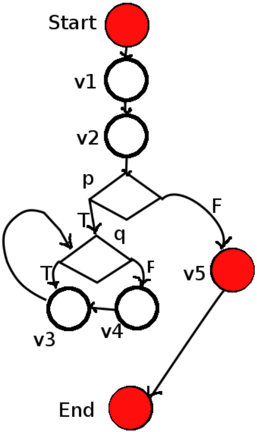
Traditional Slicing using Weakly Commitment-closed Sets

- So in traditional slicing, given a slicing criterion V' we must find the minimal weakly commitment closed set containing V'.
- We have an $O(n^3)log(n)$ algorithm for this. This is the same as for Amtoft's $\xrightarrow{\text{WOD}}$.

Traditional Slicing using Weakly Commitment-closed Sets

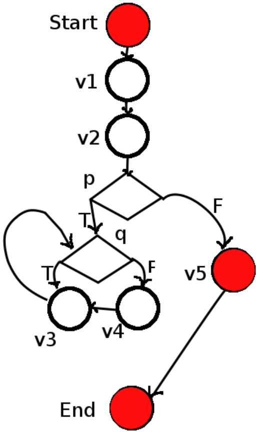
- So in traditional slicing, given a slicing criterion V' we must find the minimal weakly commitment closed set containing V'.
- We have an $O(n^3)log(n)$ algorithm for this. This is the same as for Amtoft's $\xrightarrow{\text{WOD}}$.
- We believe it can be improved to $O(n^3)$.

Another Example



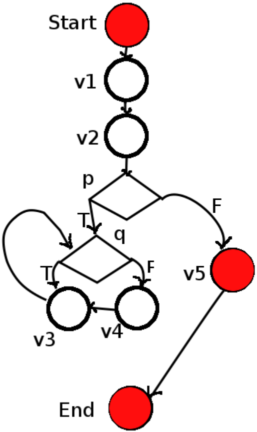
This set is weakly commitment-closed.

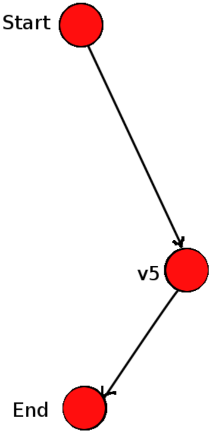
Another Example



This set is weakly commitment-closed. What is the induced graph?

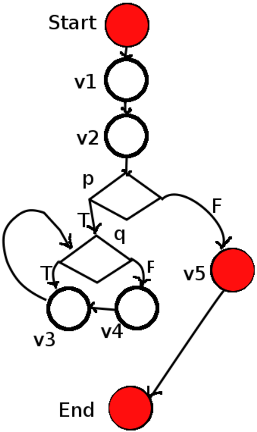
Another Example

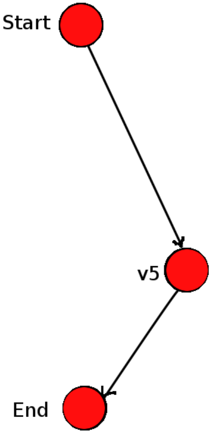




This set is weakly commitment-closed. This is the induced graph.

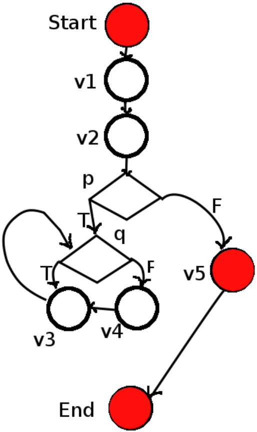
Another Example

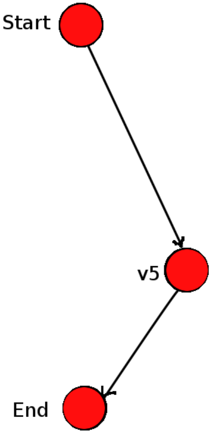




Any comments regarding non-termination?

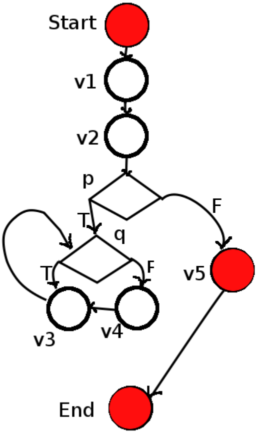
WCC does not preserve non-termination

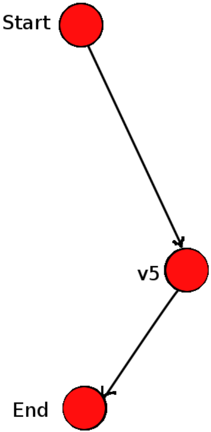




It certainly does not preserve non-termination.

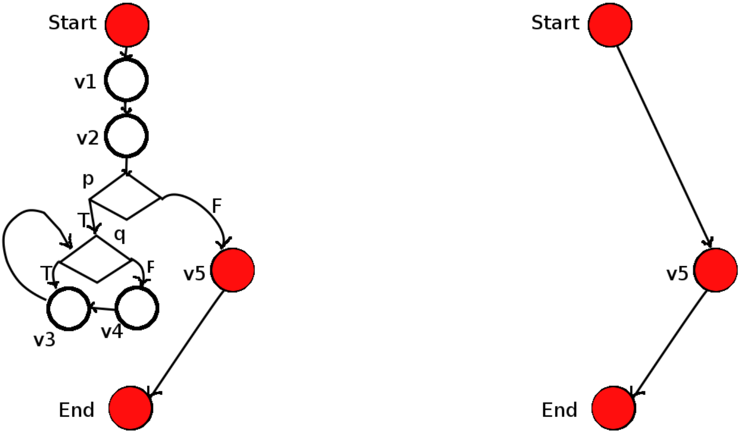
WCC does not preserve non-termination





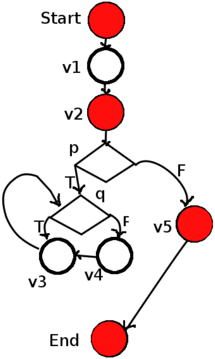
It certainly does not preserve non-termination. But that's not surprising because this is **weak** commitment-closedness.

We need Strong Commitment Closedness for that.



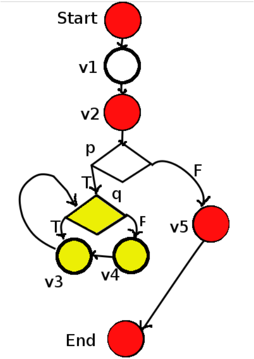
To preserve non-termination we need **strong** commitment closedness.

S-avoiding Nodes



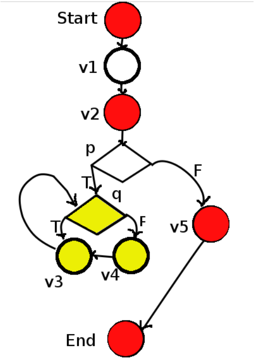
A node is S-**avoiding** if no paths from it reach S.

S-avoiding Nodes



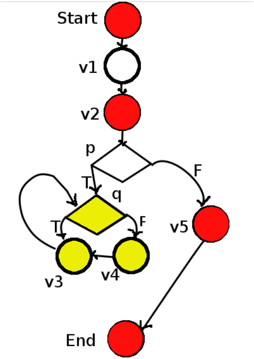
A node is S-**avoiding** if no paths from it reach S. q, v_3, v_4 are S-avoiding .

S-Strongly Committing Nodes



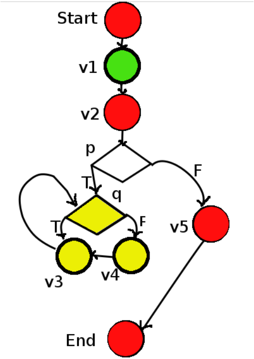
A node is S-**strongly committing** if it is S-weakly committing and all paths from it eventually reach S.

S-Strongly Committing Nodes



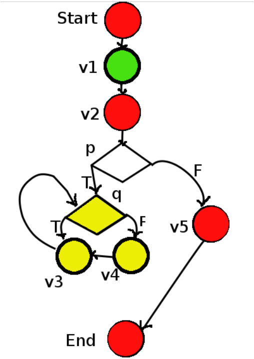
A node is S-**strongly committing** if it is S-weakly committing and all paths from it eventually reach S. i.e. all paths from it reach the same element of S first.

S-Strongly Committing Nodes



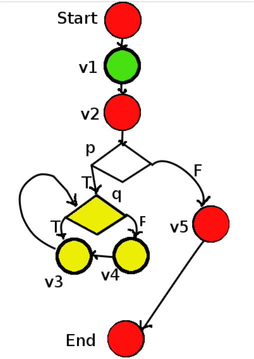
A node is S-**strongly committing** if it is S-weakly committing and all paths from it eventually reach S. i.e. all paths from it reach the same element of S first. v_1 is S-strongly committing.

Strong Commitment Closedness



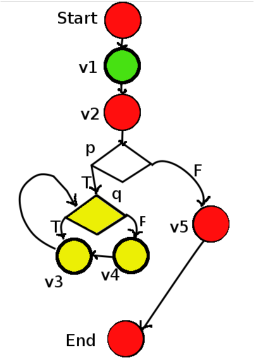
S is **strongly commitment-closed** if all elements not in S are either S-avoiding or S-strongly committing.

Strong Commitment Closedness



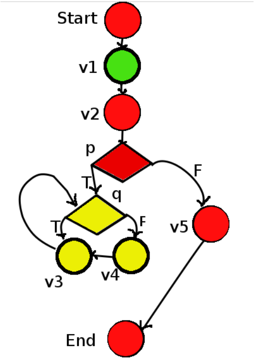
S is **strongly commitment-closed** if all elements not in S are either S-avoiding or S-strongly committing. p is neither S-avoiding nor S-strongly committing.

Strong Commitment Closedness



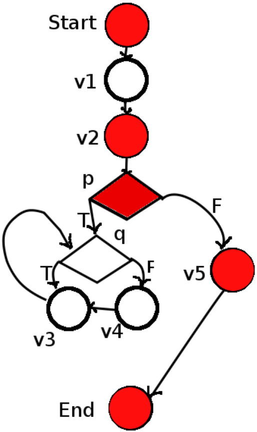
S is **strongly commitment-closed** if all elements not in S are either S-avoiding or S-strongly committing. p is neither S-avoiding nor S-strongly committing. So S is not strongly commitment-closed.

Strong Commitment Closedness



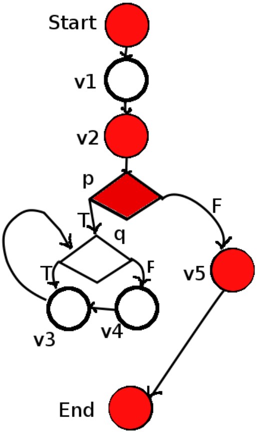
S is **strongly commitment-closed** if all elements not in S are either S-avoiding or S-strongly committing. p is neither S-avoiding nor S-strongly committing. So S is not strongly commitment-closed. Now it is!

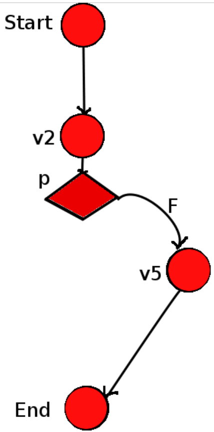
Graphs Induced from Strongly Commitment Closed Sets



So let's look at this induced graph.

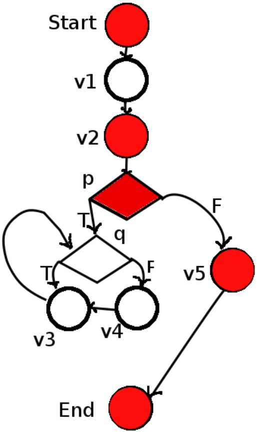
Graphs Induced from Strongly Commitment Closed Sets

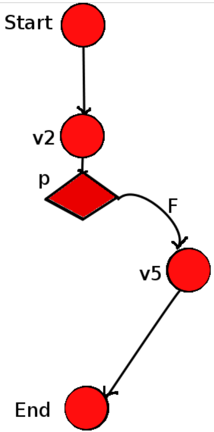




So let's look at this induced graph.

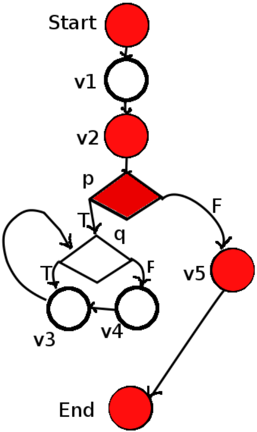
Incomplete Predicates

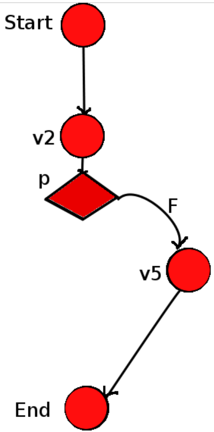




So let's look at this induced graph. p is an 'incomplete' predicate.

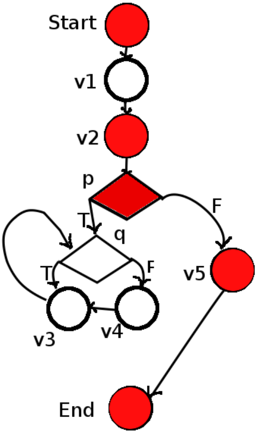
Interpreting Incomplete Predicates

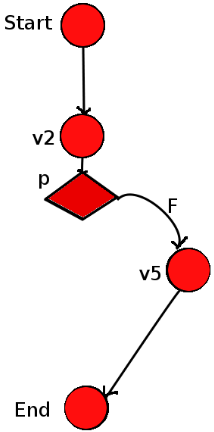




So let's look at this induced graph. p is an 'incomplete' predicate. How do we interpret this?

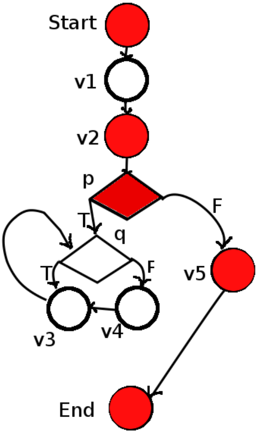
Interpreting Incomplete Predicates

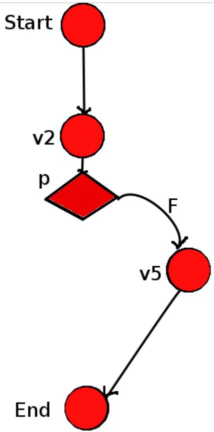




So let's look at this induced graph. p is an 'incomplete' predicate. How do we interpret this? If p evaluates to T then we get silent non-termination.

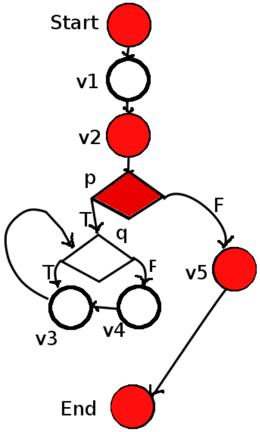
The Advantage of Incomplete Predicates





Using incomplete predicates for silent non-termination means that we don't have to include 'ghost' control sinks that may introduce further unnecessary dependences.

Theorem 2: Soundness and Completeness of SCC



For each form c of the strong forms of control dependence in the literature, S is closed under c if and only if S is strongly commitment-closed.

Non-termination Sensitive Slicing using Strongly Commitment-closed Sets

- So in Non-termination Sensitive Slicing slicing, given a slicing criterion V' we must find the minimal strongly commitment-closed set containing V'.

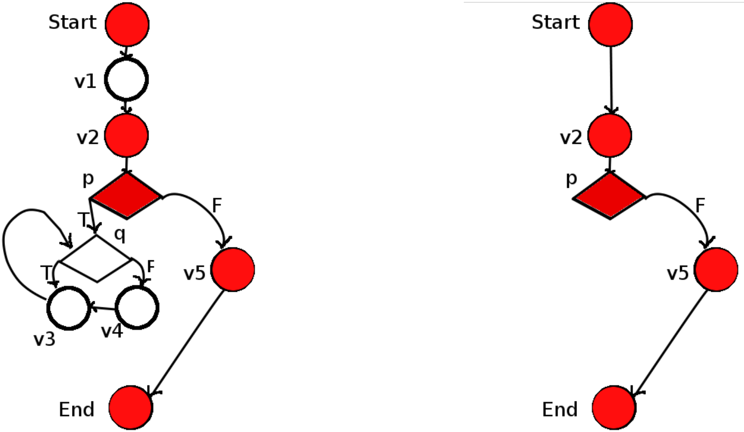
Non-termination Sensitive Slicing using Strongly Commitment-closed Sets

- So in Non-termination Sensitive Slicing slicing, given a slicing criterion V' we must find the minimal strongly commitment-closed set containing V'.
- Again, we have an $O(n^3)log(n)$ algorithm for this.

Non-termination Sensitive Slicing using Strongly Commitment-closed Sets

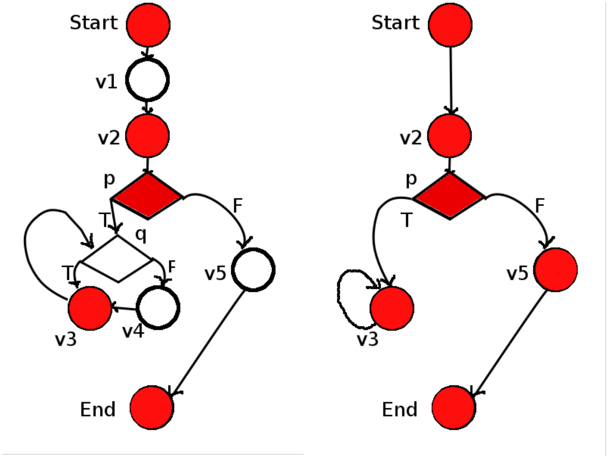
- So in Non-termination Sensitive Slicing slicing, given a slicing criterion V' we must find the minimal strongly commitment-closed set containing V'.
- Again, we have an $O(n^3)log(n)$ algorithm for this.
- Again, we believe it can be improved to $O(n^3)$.

Semantics?



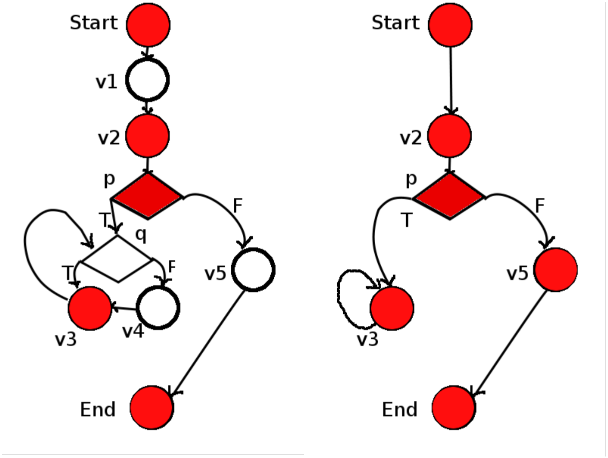
What is the semantic relationship between a graph and the graphs induced by a weakly and strongly commitment-closed sets?

Semantics Induced by Weakly Commitment-closed Sets



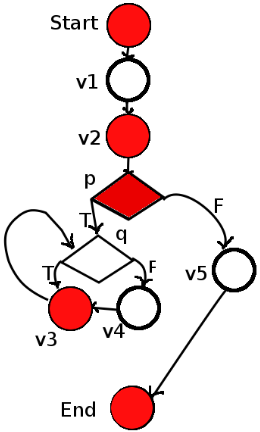
What is the semantic relationship between a graph and a graph induced by a weakly commitment-closed set?

Walks



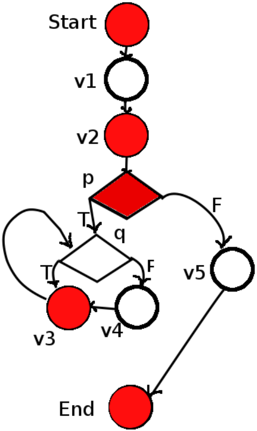
Walks are like paths where we also record whether the T or F branches were taken at the predicates.

Examples of Walks



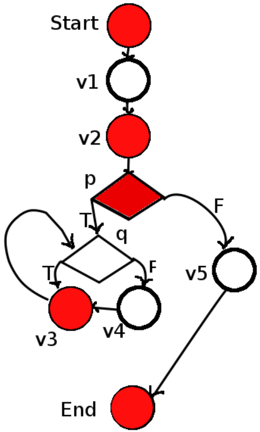
start, v_1, v_2

Examples of Walks



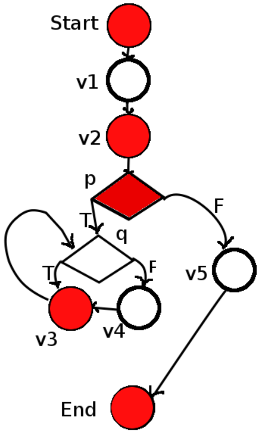
$\textbf{start}, v_1, v_2, (p, \mathrm{T})$

Examples of Walks



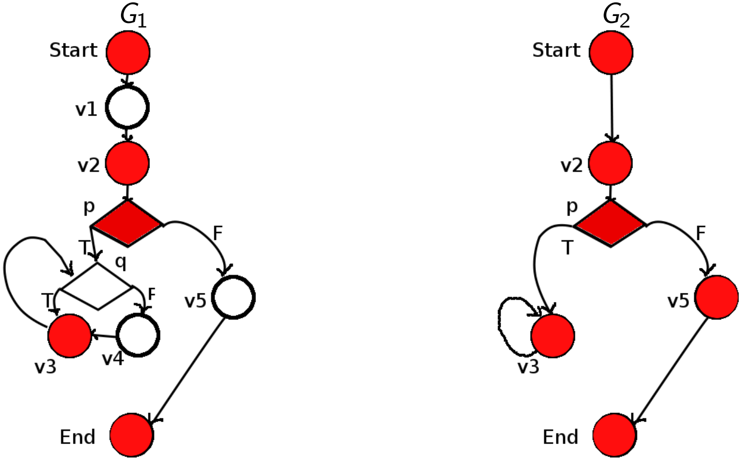
start, v_1, v_2, (p, T), (q, F), v_4, v_3, (q, T), v_3

Examples of Walks



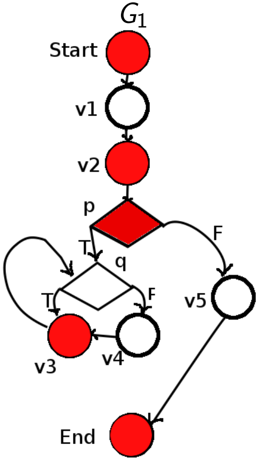
$\mathbf{start}, v_1, v_2, (p, \mathrm{F}), v_5, \mathbf{end}$

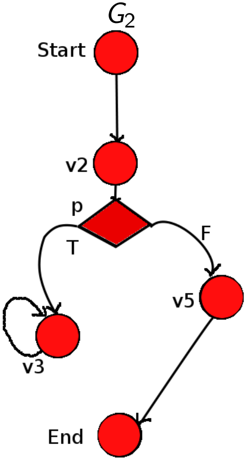
Walks of the Induced Graph



Let's compare walks of the original graph with the walks of a graph induced by a weakly commitment closed set.

Walks of the Induced Graph

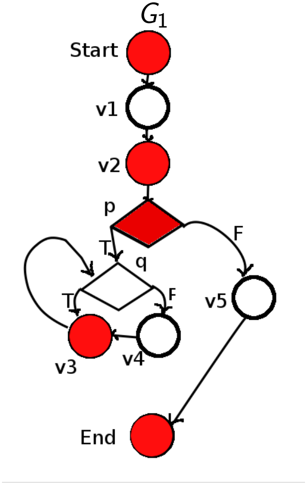


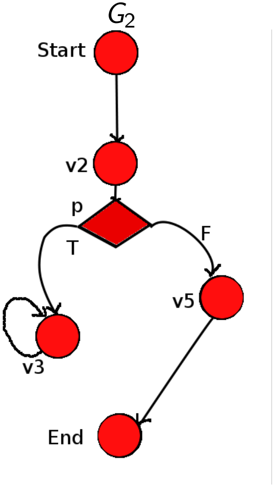


start, v_1, v_2

start, v_2

Walks of the Induced Graph

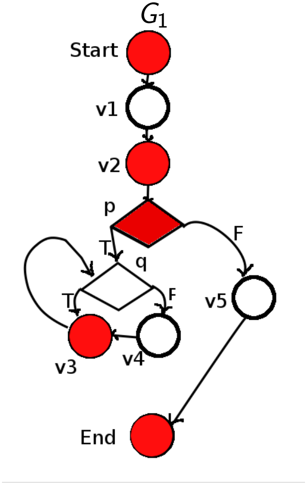


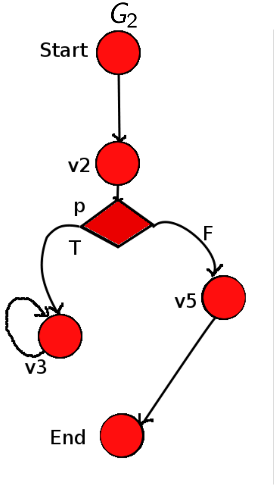


start, v_1, v_2, (p, T)

start, v_2, (p, T)

Walks of the Induced Graph

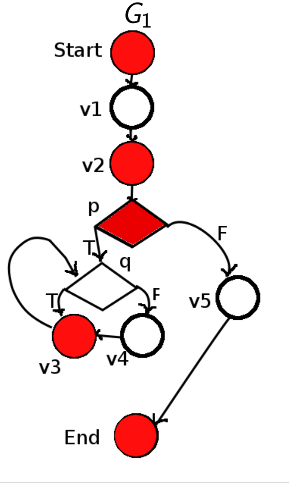


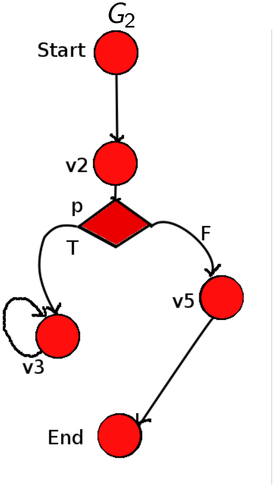


start, v_1, v_2, (p, T), (q, F), v_4, v_3, (q, T), v_3

start, v_2, (p, T), v_3, v_3

Walks of the Induced Graph

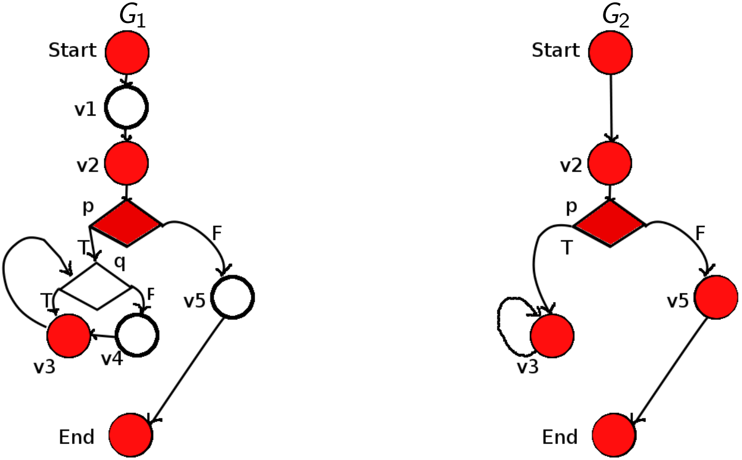




start, v_1, v_2, (p, F), v_5, **end**

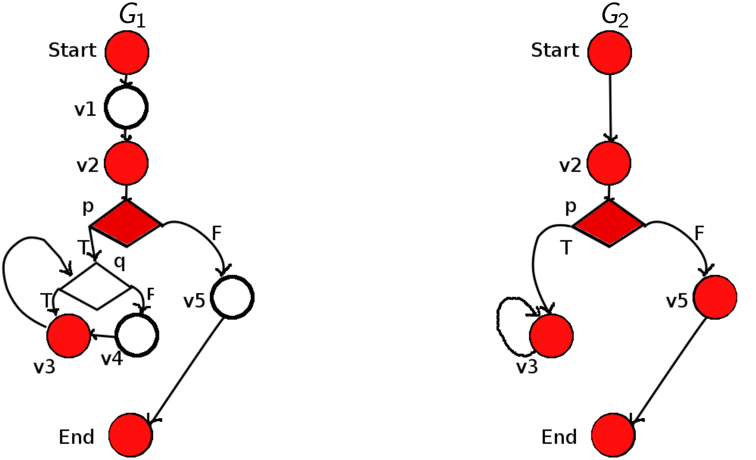
start, v_2, (p, F), v_5, **end**

Walks of Graphs Induced from WCC Sets



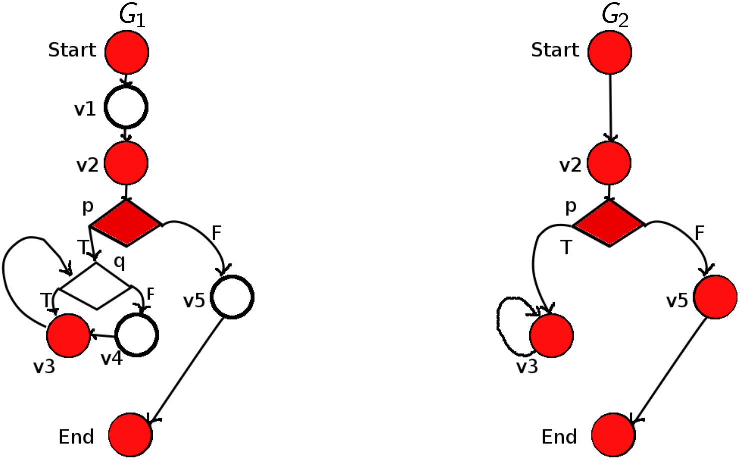
What is the relationship between the walks of G_1 and the walks of G_2?

Weak Projections



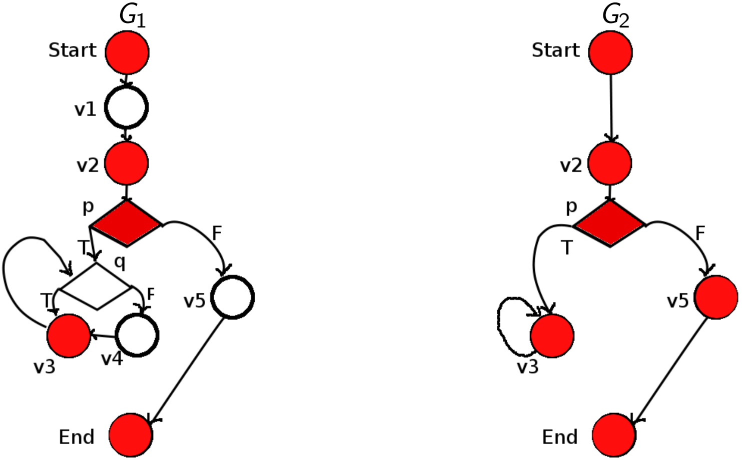
Every walk of G_1 when restricted to G_2 is a walk of G_2.

Weak Projections



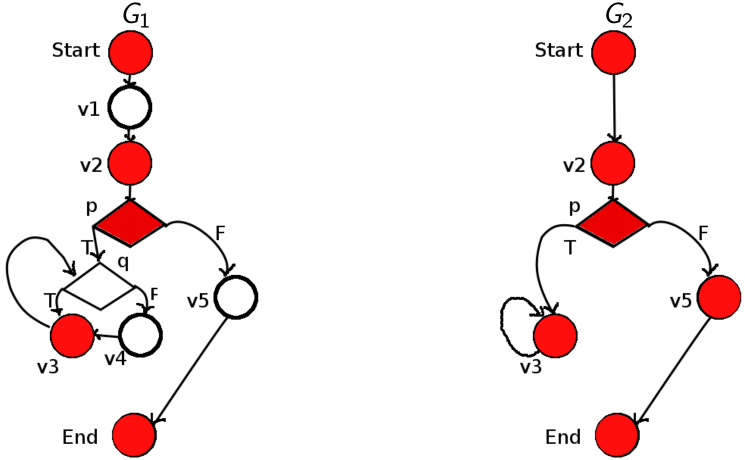
Every walk of G_1 when restricted to G_2 is a walk of G_2. We say G_2 is a **weak projection** of G_1.

Theorem 3: Semantics of WCC



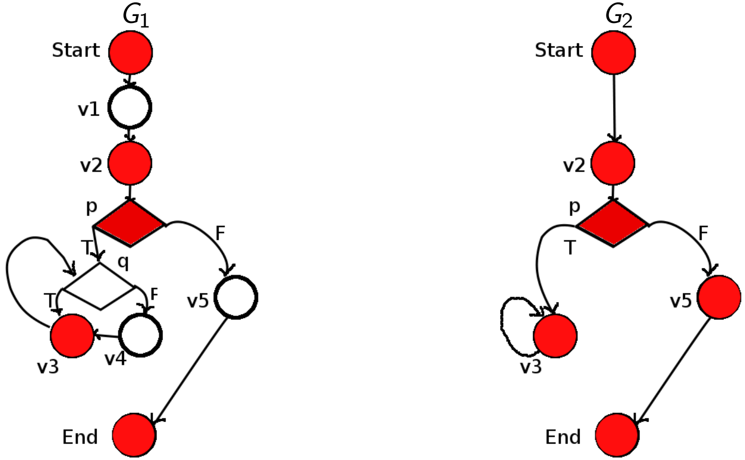
The graph induced from V is a weak projection if and only if V is weakly commitment-closed.

Result



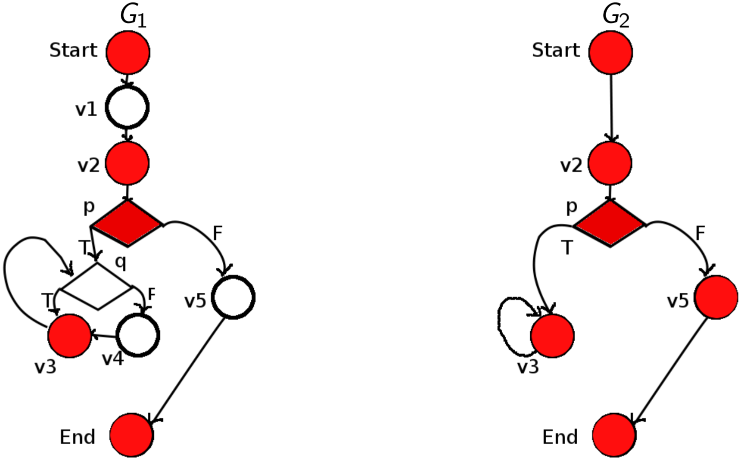
Theorems 1 and 3 imply that sets closed under all weak forms of control dependence in the literature induce weak projections.

Weak Projections



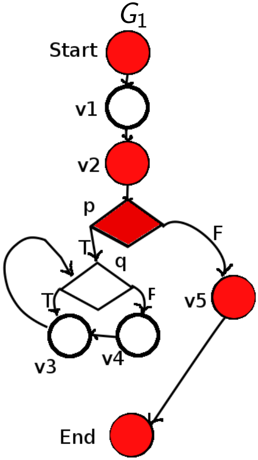
So weak projection captures semantically what all previous authors of definitions of weak control dependence wanted to achieve!

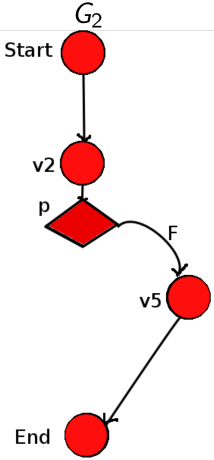
Weak Control Dependence



So authors of future definitions should also prove their definitions satisfy this property!

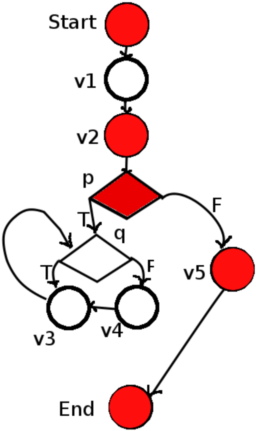
Strong Control Dependence

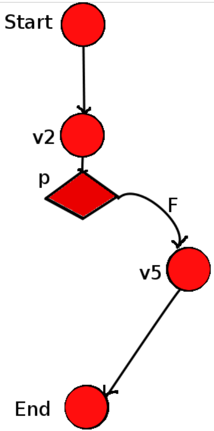




What about graphs induced from strongly commitment-closed sets?

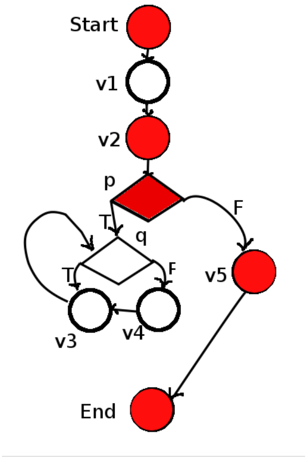
Maximal Walks

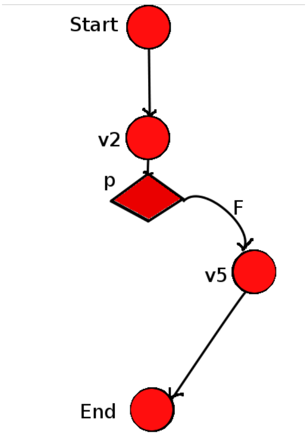




Maximal walks are those which are not a prefix of any other walk.

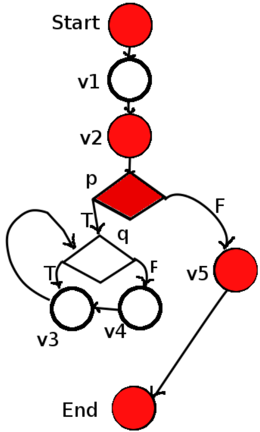
Maximal Walks corresponding to termination

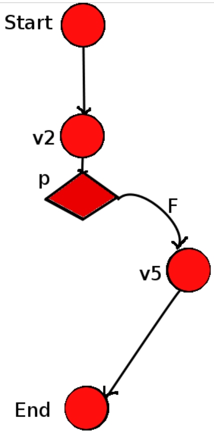




The only maximal walks that correspond to termination are those whose final element is **end**.

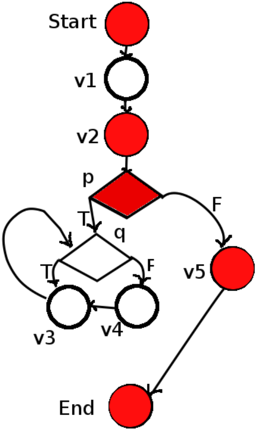
Maximal Walks corresponding to non-termination

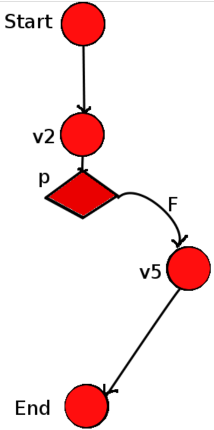




All other finite maximal walks are considered non-terminating.

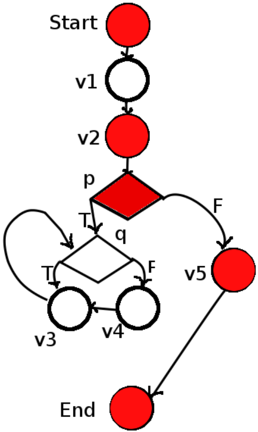
Walks of Graphs Induced from SCC sets

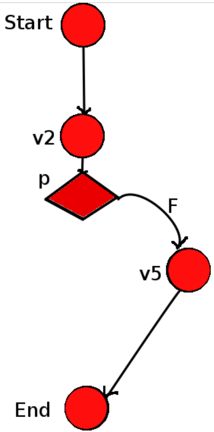




start, v_1, v_2, (p, T), (q, T), v_3, (q, F), v_4, $v_3 \ldots$

Walks of Graphs Induced from SCC sets

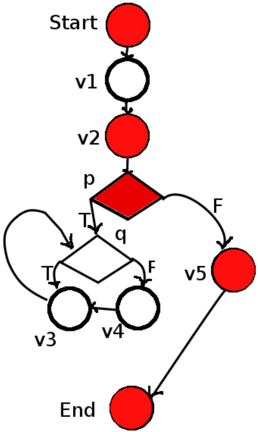


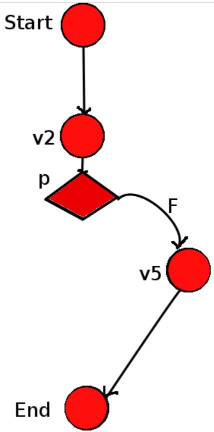


$\textbf{start}, v_1, v_2, (p, \mathrm{T}), (q, \mathrm{T}), v_3, (q, \mathrm{F}), v_4, v_3 \ldots$

$\textbf{start}, v_2, (p, \mathrm{T})$

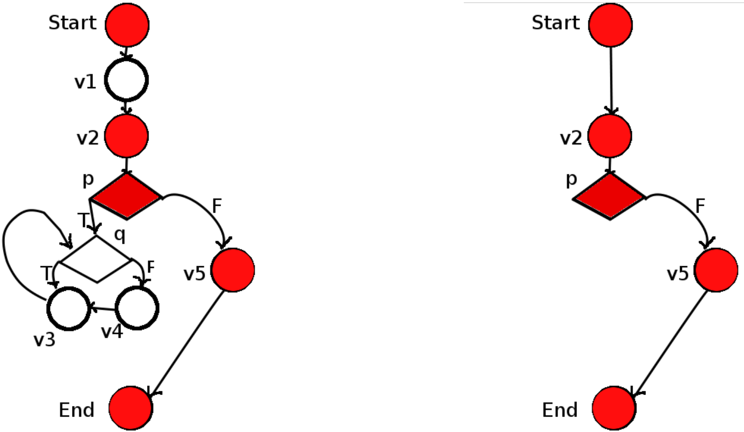
Walks of Graphs Induced from SCC sets





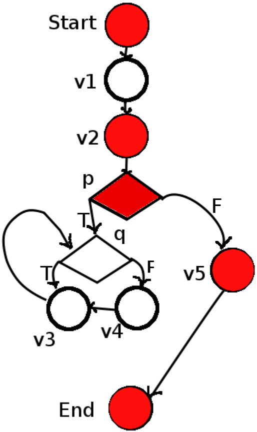
What is the relationship between the two?

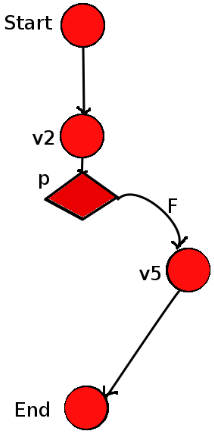
Remember Weak Projections



G_2 is a weak projection of G_1 means every walk of G_1 when restricted to G_2 is a walk of G_2.

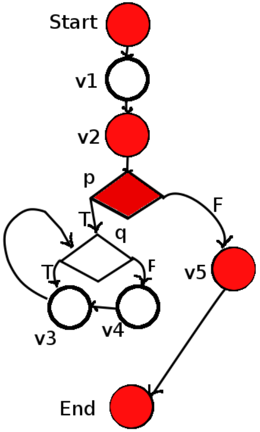
Strong Projections

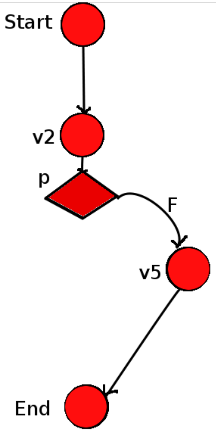




G_2 is a strong projection of G_1 means every **maximal** walk of G_1 when restricted to G_2 is a **maximal** walk of G_2.

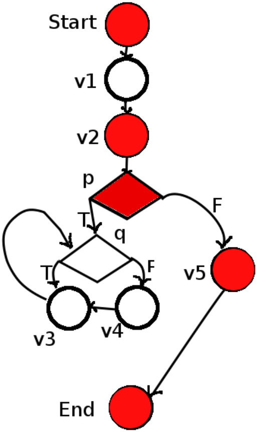
Theorem 4: Semantics of SCC

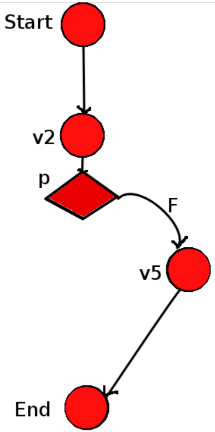




The graph induced from V is a strong projection if and only if V is strongly commitment-closed.

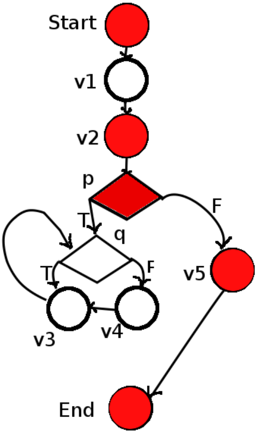
Result

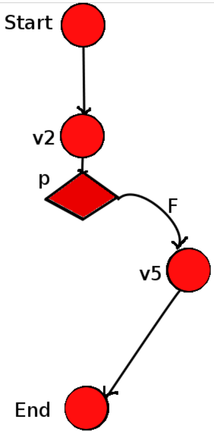




Theorems 2 and 4 imply that sets closed under all strong forms of control dependence in the literature induce strong projections.

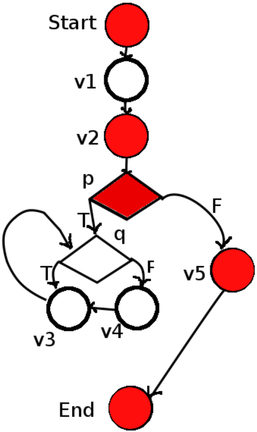
Strong Projection

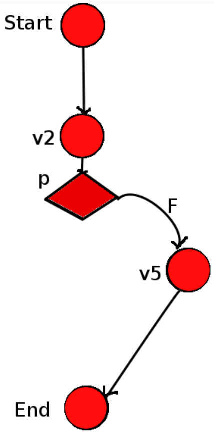




So, again, strong projection captures semantically what all previous authors of definitions of strong control dependence wanted to achieve.

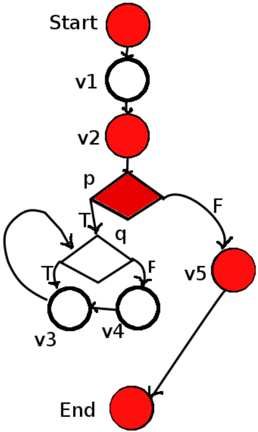
Strong Projections are Non-Termination Preserving

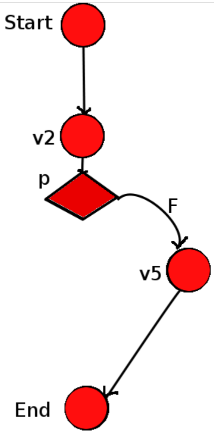




It follows that strong projections are non-termination preserving.

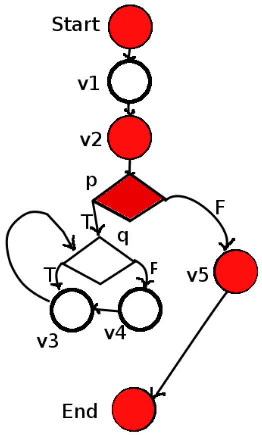
Strong Projections are Non-Termination Preserving

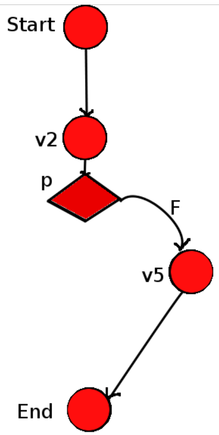




From this it follows that strong projections are non-termination preserving (as required!).

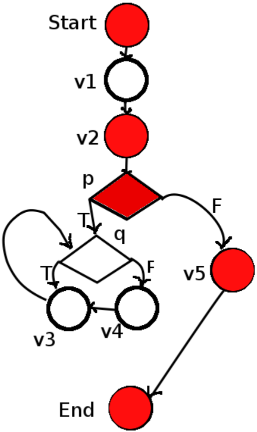
Strong Projections with **end** preserve both

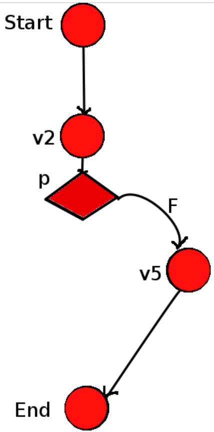




Also notice, strong projections are weak projections.

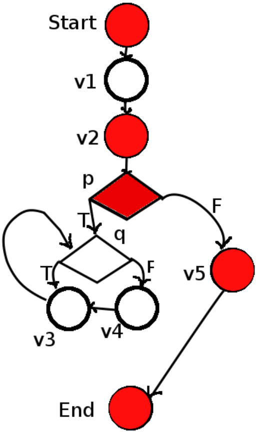
Strong Projections with **end** preserve both

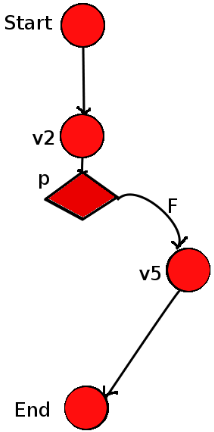




Also notice, strong projections are weak projections. But not vice-versa.

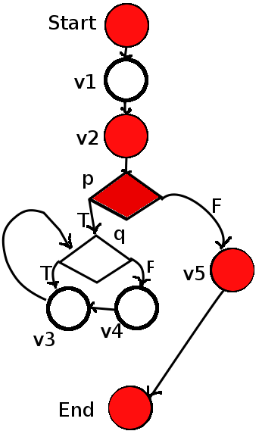
Strong Projections with **end** preserve both

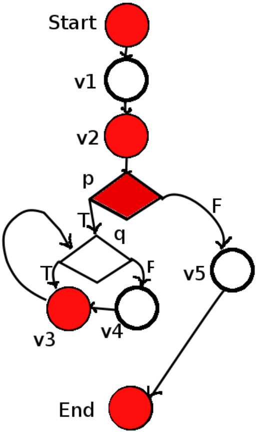




If **end** is in the weak projection, then the weak projection preserves termination.

Strong Projections with **end** preserve both





So if **end** is in the strong projection, then the strong projection preserves both termination and non-termination.

Conclusion

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

Conclusion

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way?

Conclusion

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way? yes!

Conclusion

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way? yes!
- Are there underlying semantic properties captured by all these different forms of control dependence?

Conclusion

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Can they be generalised in a nice high-level way? yes!
- Are there underlying semantic properties captured by all these different forms of control dependence? yes!

Conclusion

- Weak (Non-termination sensitive):

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

Conclusion

- Weak (Non-termination sensitive):

$\xrightarrow{\text{W-controls}}$	(Weiser 1979)
$\xrightarrow{\text{F-controls}}$	(Ferrante and Ottenstein 1987)
$\xrightarrow{\text{WOD}}$	(Amtoft 2007)

- Strong (Non-termination sensitive):

$\xrightarrow{\text{PC-weak}}$	(Podgurski and Clarke 1990)
$\xrightarrow{\mathrm{NTSCD}}$ and $\xrightarrow{\text{DOD}}$	(Ranganath et al 2006)

Conclusions

Conclusions

- We have generalised non-termination insensitive and sensitive control dependence by defining and giving algorithms for **weak** and **strong commitment-closedness**.

Conclusions

- We have generalised non-termination insensitive and sensitive control dependence by defining and giving algorithms for **weak** and **strong commitment-closedness**.
- Theorem 1: A set is closed under each of the weak forms of control dependence in the literature if and only if it is weakly commitment-closed.

Conclusions

- We have generalised non-termination insensitive and sensitive control dependence by defining and giving algorithms for **weak** and **strong commitment-closedness**.
- Theorem 1: A set is closed under each of the weak forms of control dependence in the literature if and only if it is weakly commitment-closed.
- Theorem 2: A set is closed under each of the strong forms of control dependence in the literature if and only if it is strongly commitment-closed.

Conclusions

- We have generalised non-termination insensitive and sensitive control dependence by defining and giving algorithms for **weak** and **strong commitment-closedness**.
- Theorem 1: A set is closed under each of the weak forms of control dependence in the literature if and only if it is weakly commitment-closed.
- Theorem 2: A set is closed under each of the strong forms of control dependence in the literature if and only if it is strongly commitment-closed.
- We defined **semantic** relations: **weak** and **strong projections** between graphs in terms of **walks**.

Conclusions

- We have generalised non-termination insensitive and sensitive control dependence by defining and giving algorithms for **weak** and **strong commitment-closedness**.
- Theorem 1: A set is closed under each of the weak forms of control dependence in the literature if and only if it is weakly commitment-closed.
- Theorem 2: A set is closed under each of the strong forms of control dependence in the literature if and only if it is strongly commitment-closed.
- We defined **semantic** relations: **weak** and **strong projections** between graphs in terms of **walks**.
- Theorem 3: The graph induced from V is a weak projection if and only if V is weakly commitment-closed.

Conclusions

- We have generalised non-termination insensitive and sensitive control dependence by defining and giving algorithms for **weak** and **strong commitment-closedness**.
- Theorem 1: A set is closed under each of the weak forms of control dependence in the literature if and only if it is weakly commitment-closed.
- Theorem 2: A set is closed under each of the strong forms of control dependence in the literature if and only if it is strongly commitment-closed.
- We defined **semantic** relations: **weak** and **strong projections** between graphs in terms of **walks**.
- Theorem 3: The graph induced from V is a weak projection if and only if V is weakly commitment-closed.
- Theorem 4: The graph induced from V is a strong projection if and only if V is strongly commitment-closed.

The End

Thanks for listening?

The End

Thanks for listening?

Any questions?