Critical Expansions Methodology at the Service of Formal Argumentation

Dov Gabbay
King's College London and University of Luxembourg

November 25, 2020

636-Dov-Jeremie-Slides

Abstract

The second half of the previous century has seen a meteoric rise in the landscape of logical systems trying to model applications in computer science, AI, language analysis, KR, formal philosophy and more.
Fortunately, some methodological order and evaluation could be introduced using Classical Logic and its proof theories.
In this century there was the rise of many formal argumentation systems and semantics addressing a variety of applications, as well as absorbing many aspects of the new logics of the previous century.

Abstract: continued

There is now the need for a methodology to play the role/provide a home for comparison and evaluation, a role similar to that of classical logic in the previous century. We offer the methodology of critical expansions.
These slides present a qualitative analysis of how the idea of critical expansions can be used for resolving loops/cycles in formal argumentation networks.
Let us begin.

A generic enhanced argumentation network can be presented schematically in the form $(S, R,$ **Enhancements**$)$, where S is the set of arguments arising/mirroring from some intended application area, R is the basic binary attack relation on S and **Enhancements** is the various additional aspect coming from the application.
A critical expansion for $(S, R,$ **Enhancements**$)$ has the form (S, S^*, R^*), where S^* enlarges S with additional service arguments, and where R^* is the attack relation on S^*, devised in such a way that (S^*, R^*) is a critical faithful expansion of $(S, R,$ **Enhancements**$)$ representing the enhancements of $(S, R,$ **Enhancements**$)$ by means of the service arguments.
The enhancements are implemented by the way S^* interacts with S, using R^*.

The lecture will illustrate this idea (using examples) for higher level attacks (any level) and for resolution of cycles.

- Presenting the critical Control–Execute Expansion (slides 4–7)
- The many faces of a sample network (slides 8–13)
- Methodological discussion (slides 14–end)

Part 1: The control execute expansion explained.

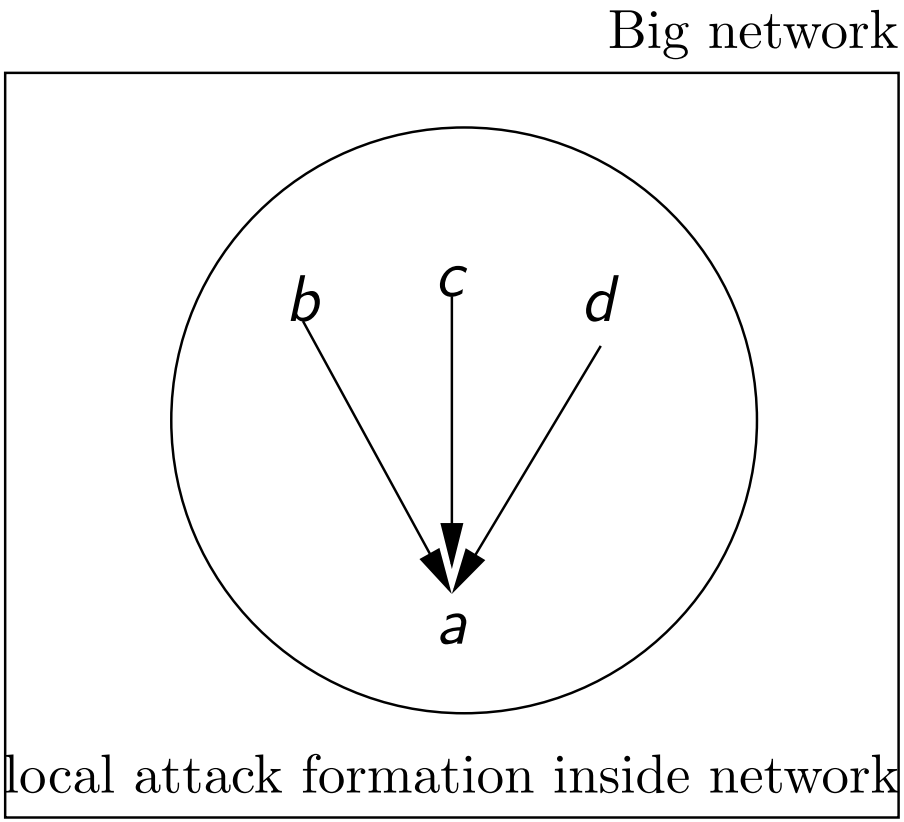


Figure: 1

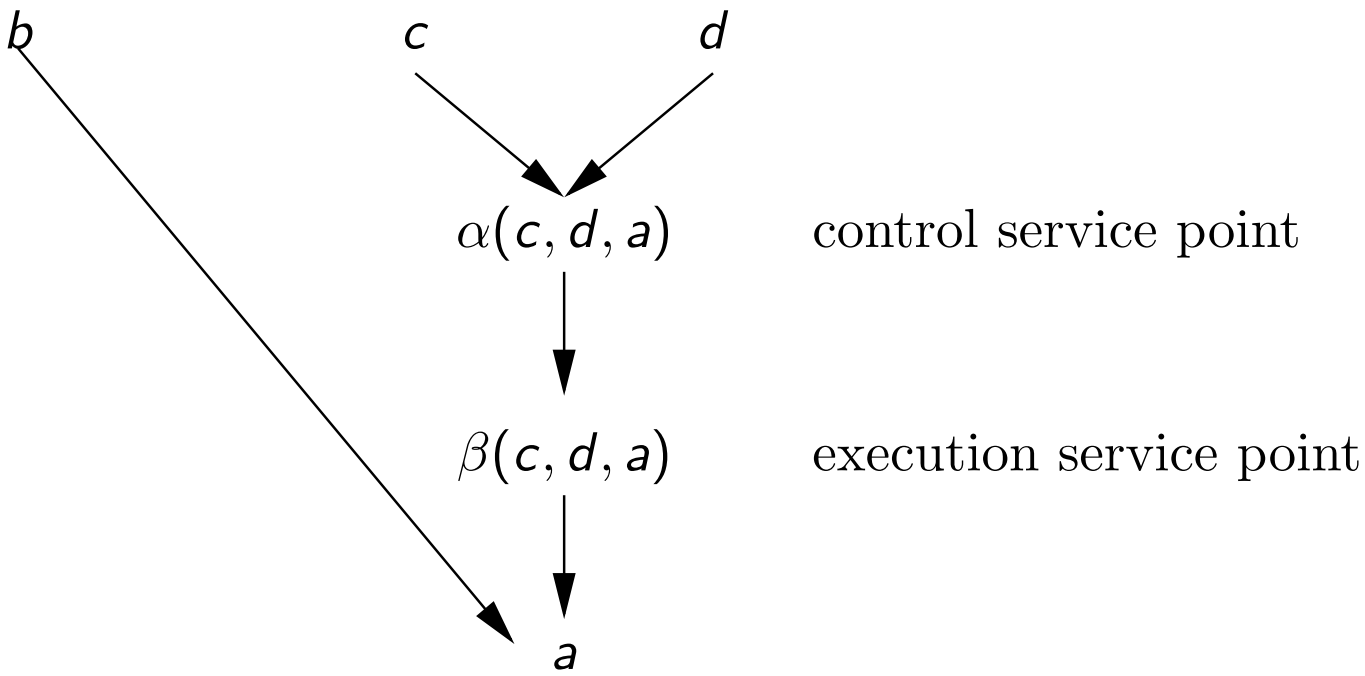


Figure: 2

Figure 2 is equivalent to Figure 1.

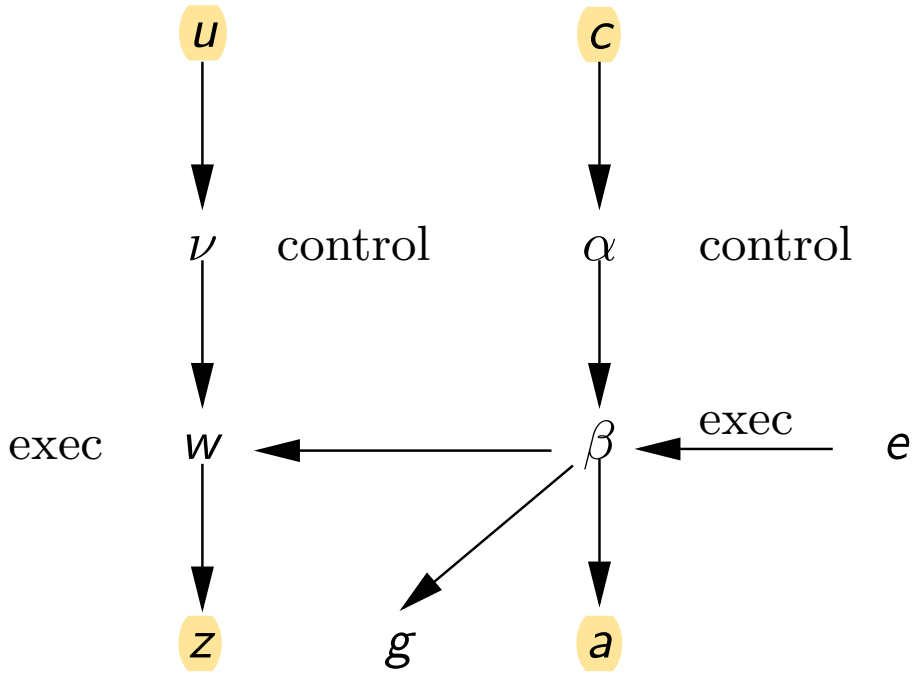


Figure: 3

We start with $c \rightarrow a$ and $u \rightarrow z$. We expand with service points as in Figure 3.

Execution point represents the attack. Control point coordinates the attackers. "$c \rightarrow a$" can be attacked through its execution point, and can launch an attack on "$u \rightarrow z$" through its execution point.
We thus get that Figure 3 is equivalent to Figure 4, (or Figure 4 is expanded into Figure 3).

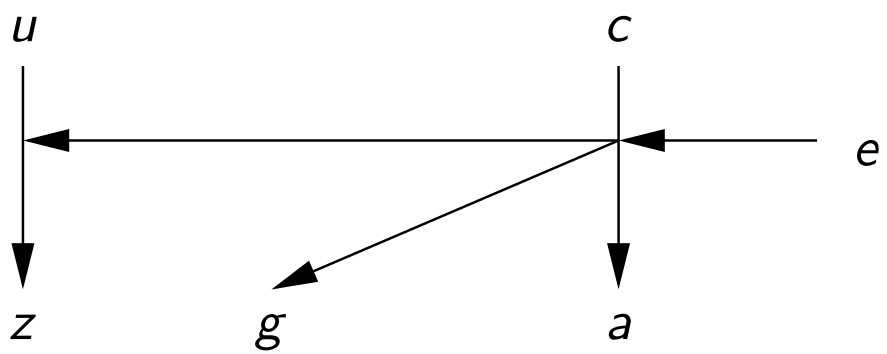


Figure: 4

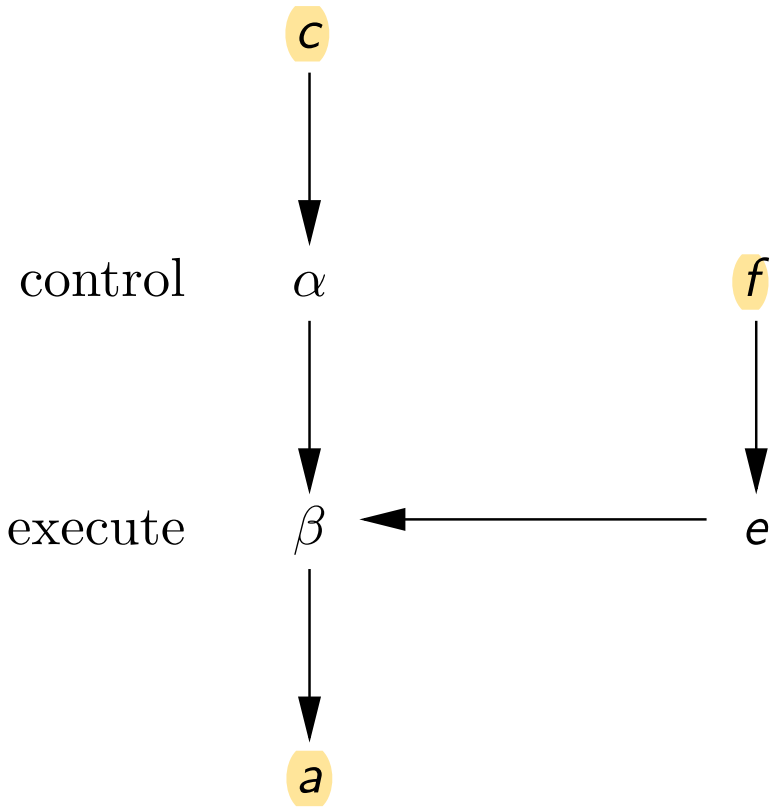


Figure: 5

Figure 5 is an expansion of Figure 6.

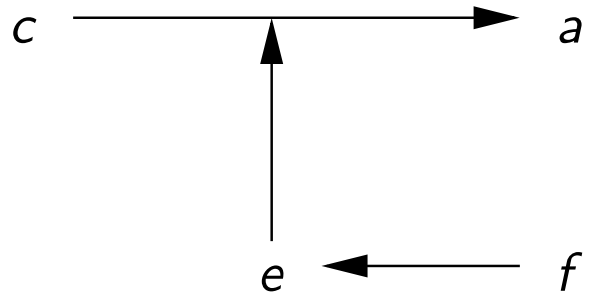


Figure: 6

e deactivates $c \rightarrow a$

f activates $c \rightarrow a$ by attacking e

So we can view c and f as jointly attacking a, as in Figure 7.

c and f jointly attack a so f can be viewed as "$c \rightarrow a$". But this view is not always correct.

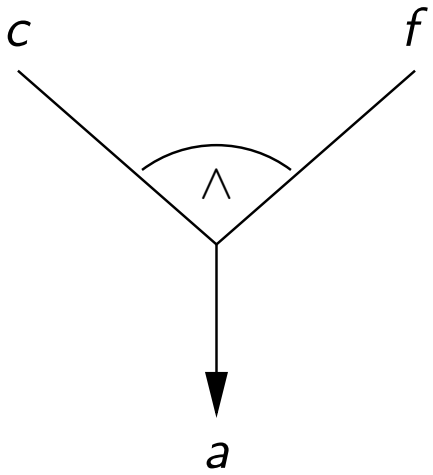


Figure: 7

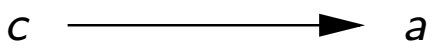


Figure: 8

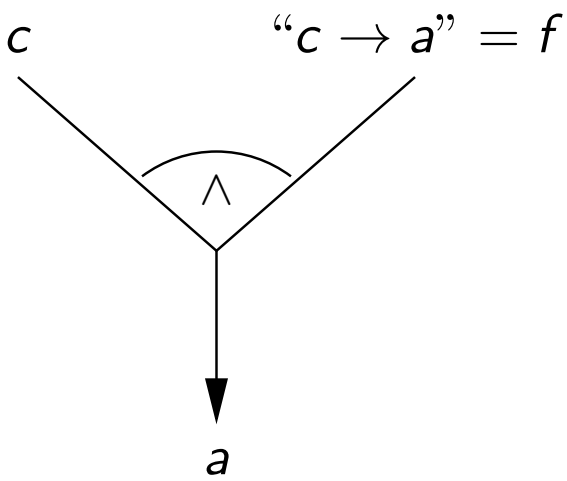


Figure: 9

Figure 8 is the same as figure 9.

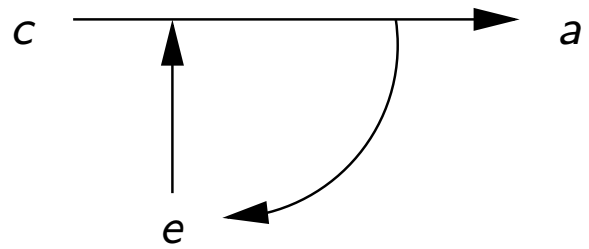


Figure: 10

Since $f =$ "$c \rightarrow a$" in Figure 9, we can present it as Figure 10.

Part 2: A Running Example

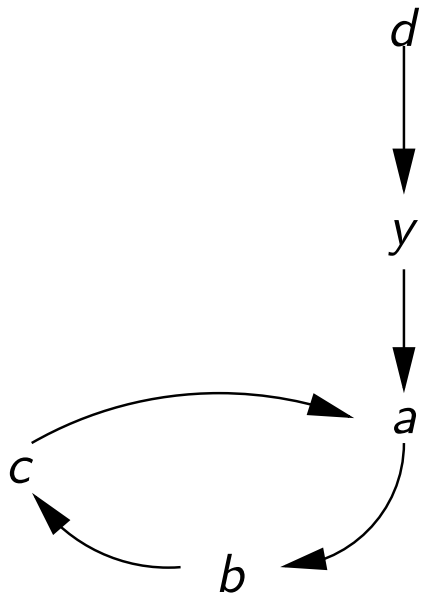


Figure: 11

The many faces of a network. Which points are service points? Note that if we do not make any choice of service points, then x is in, y is out and we are left with an all undecided three loop.

The following slides show possible choices.

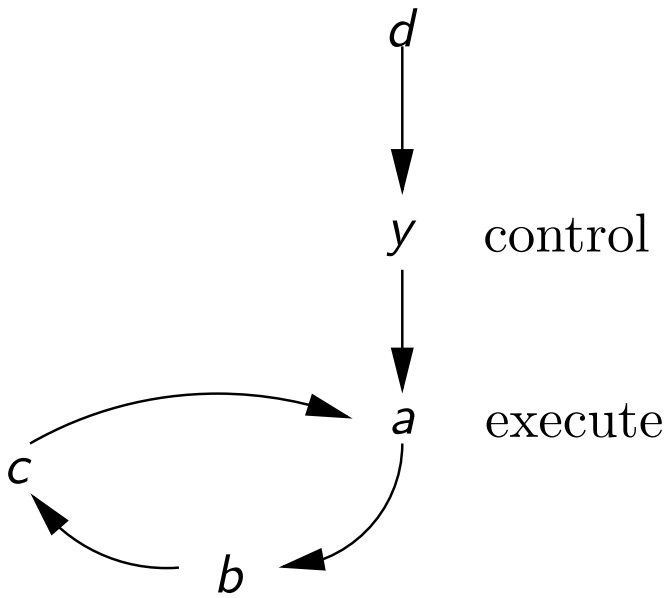


Figure: 12

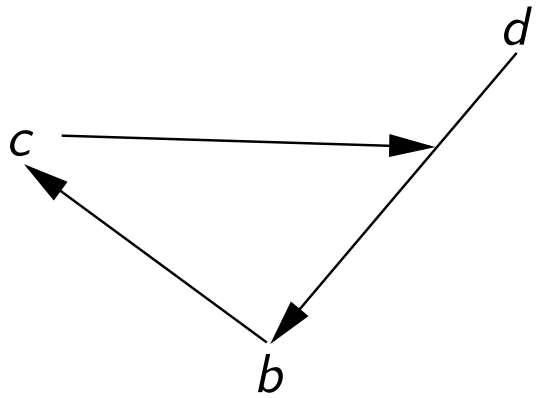


Figure: 13

For the choice in Figure 12, we get the equivalent Figure 13.

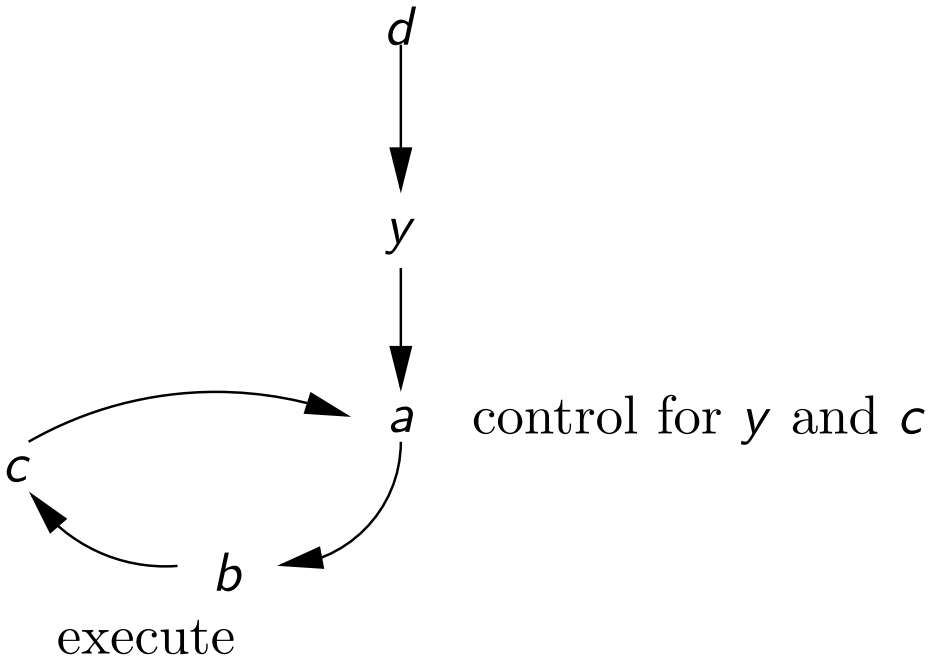


Figure: 14

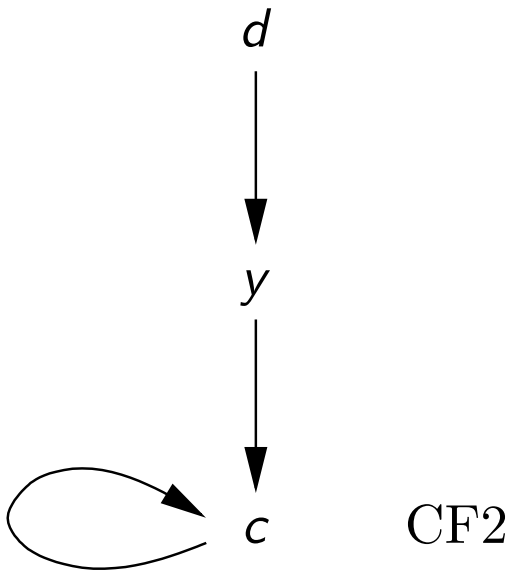


Figure: 15

For the choice in figure 14, we get the equivalent figure 15.

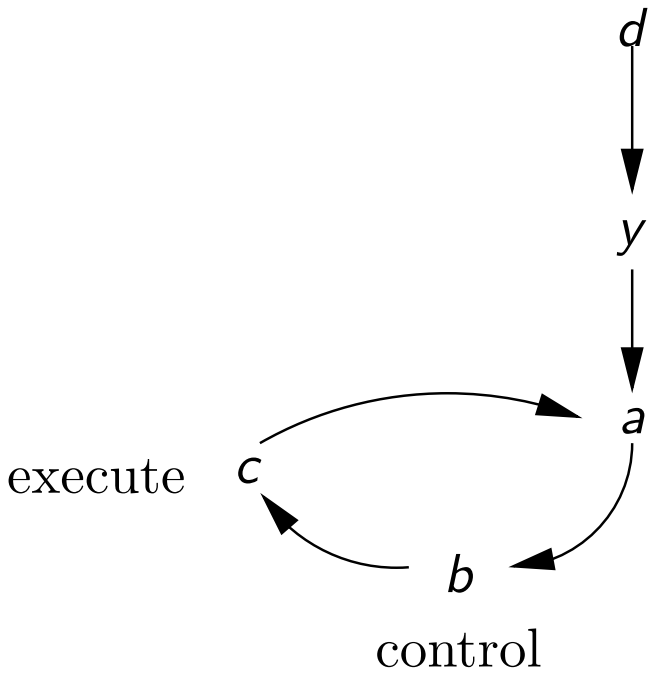


Figure: 16

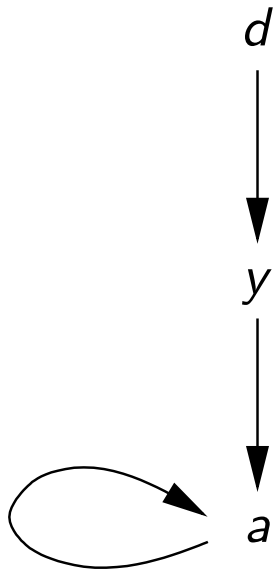


Figure: 17

For the choice in figure 16, we get the equivalent figure 17.

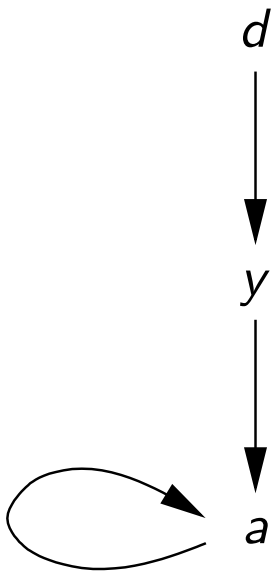


Figure: 17 copied again

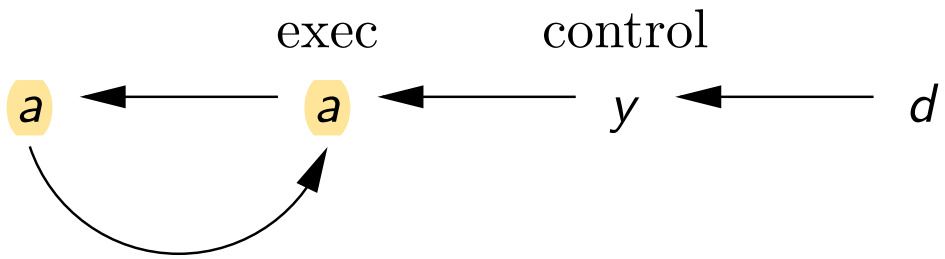


Figure: 18

Rewrite Figure 17, by repeating *a*.
Now *a* is attacking *a* (execute).

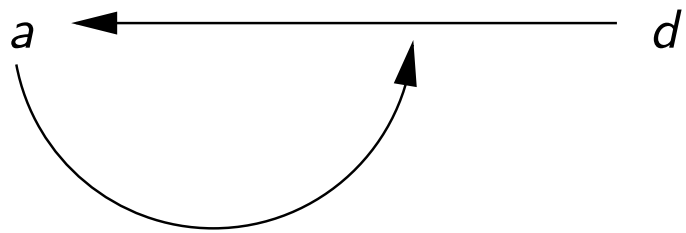


Figure: 19 - same as Figure 18

Note that Figure 19 comes from Figure 16, where we make a choice for service points.

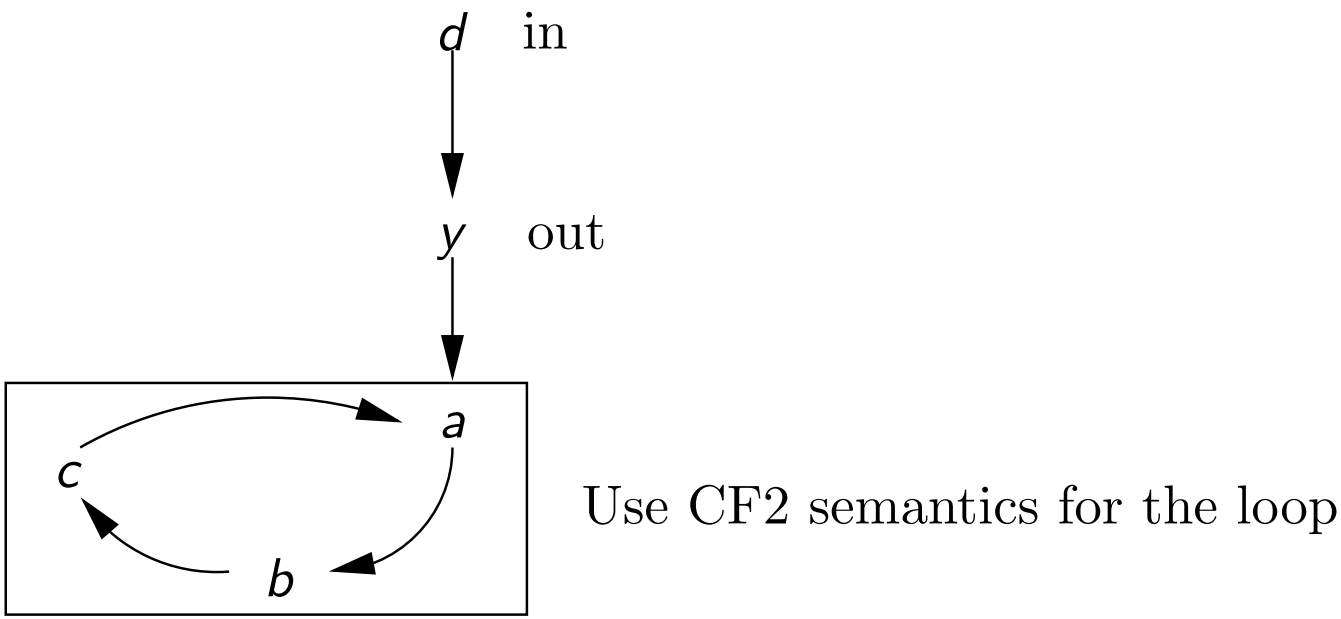


Figure: 20

Extensions computed by CF2 are $\{d, a\}, \{d, b\}, \{d, c\}$. Compare with Figure 11. In Figure 11, the traditional Dung extension is $d =$ in, $y =$ out, $a = b = c =$ undecided.

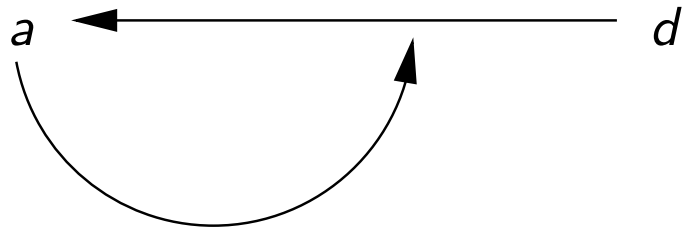


Figure: 19 - copied again

The extension in this figure is $\{d, a\}$, because d is unattacked and a can defend itself by attacking the attack "$d \rightarrow a$".
This figure was computed/rewritten from Figure 16. b was seen as control for a, and c was seen as execute for a. And Figure 16 was slowly transformed into Figure 19.

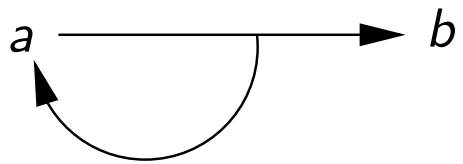


Figure: 21

Another example is Figure 21. We ask: what extensions?

1. All undecided?
2. a=out, b = in? "$(a \rightarrow b)$" =in?
3. $a = b =$ out, "$a \rightarrow b$"=in.?

Let us introduce service points.
We get Figure 22.

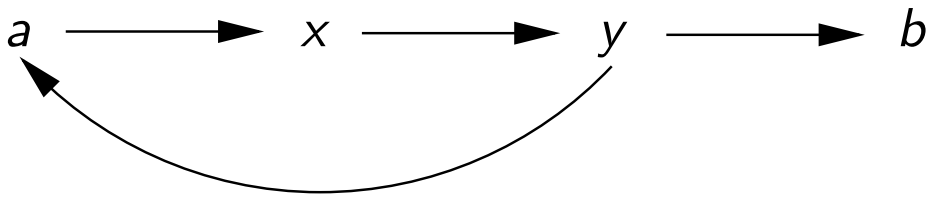


Figure: 22

Figure 22 is not so helpful in getting extensions. If we do not want to get all undecided we can use CF2 semantics or ignorer the three loop and say $b =$ in.
We try a different transformation for Figure 21.

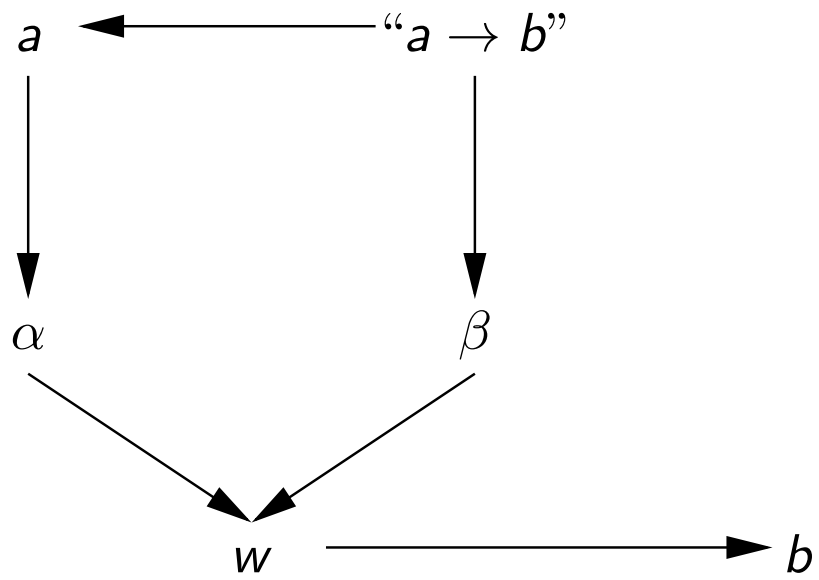


Figure: 23

Figure 21can also be transformed into Figure 23. Now we get $b =$ in, $a =$ out. b can protect itself, any attack on it actually attacks the attacker!

Problems for Gabbay's paper 636

Problem 1

Given $(S, R,$ **Enhancements**$)$, propose semantics for it!

- Eliminate all enhancements using service points. Give correct algorithm for doing this.
- Get a critical expansion (S^*, R^*). This is a traditional Dung network, but it contains clearly identifiable service points.
- Get all complete extensions of (S^*, R^*). Such extensions will contain service points.
- Give algorithms which are intuitively correct, to project the extensions thus obtained onto $(S, R,$ **Enhancements**$)$. This will give us extensions for $(S, R,$ **Enhancements**$)$.
- Prove correctness and nice properties of this method.

Problem 2

- Given a traditional Dung network (S, R) which is one big SCC loop, we seek an algorithm which identifies some elements of S as service points and thus eliminates them as exemplified in slides 8–13. This will reduce the number of points in S and give us a smaller (S', R').

  We thus get a recursive definition for eliminating loops.
- Note however that we get higher level attacks, in (S', R'). So to give semantics which eliminates loops we must give direct semantics to higher level attacks which is independent of Problem 1. We might also get an SCC which cannot be reduced further, as in Figure 36.
- Thus Problem 2 is to present such an algorithm and semantics, and evaluate its properties and applicability.

The Marcos Example

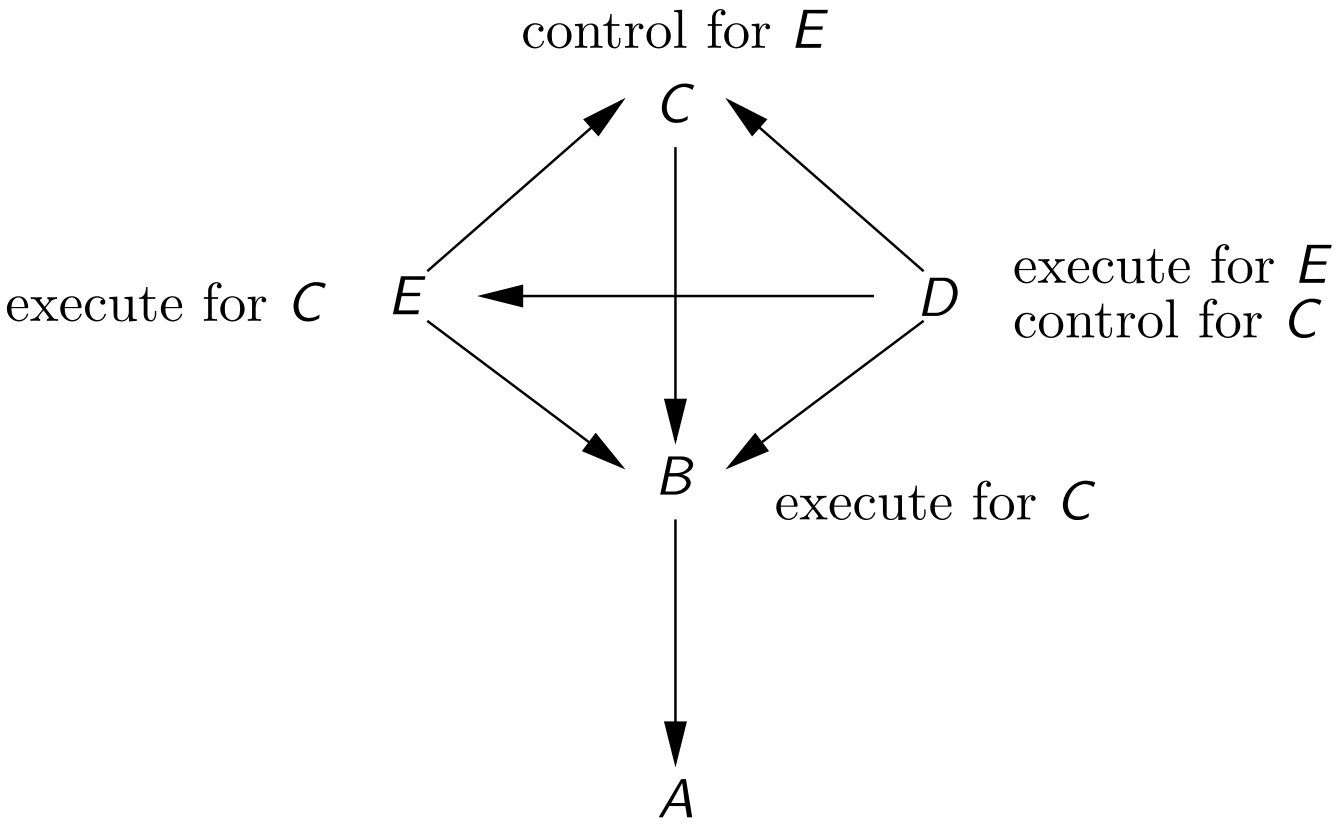


Figure: 24

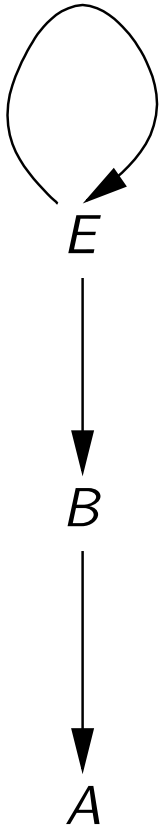


Figure: 25

Result of choosing C as control, D as execute in Figure 24.

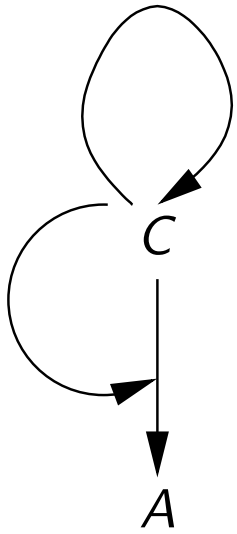


Figure: 26

Result of choosing D as control, B as execute in Figure 24.

Tim Example

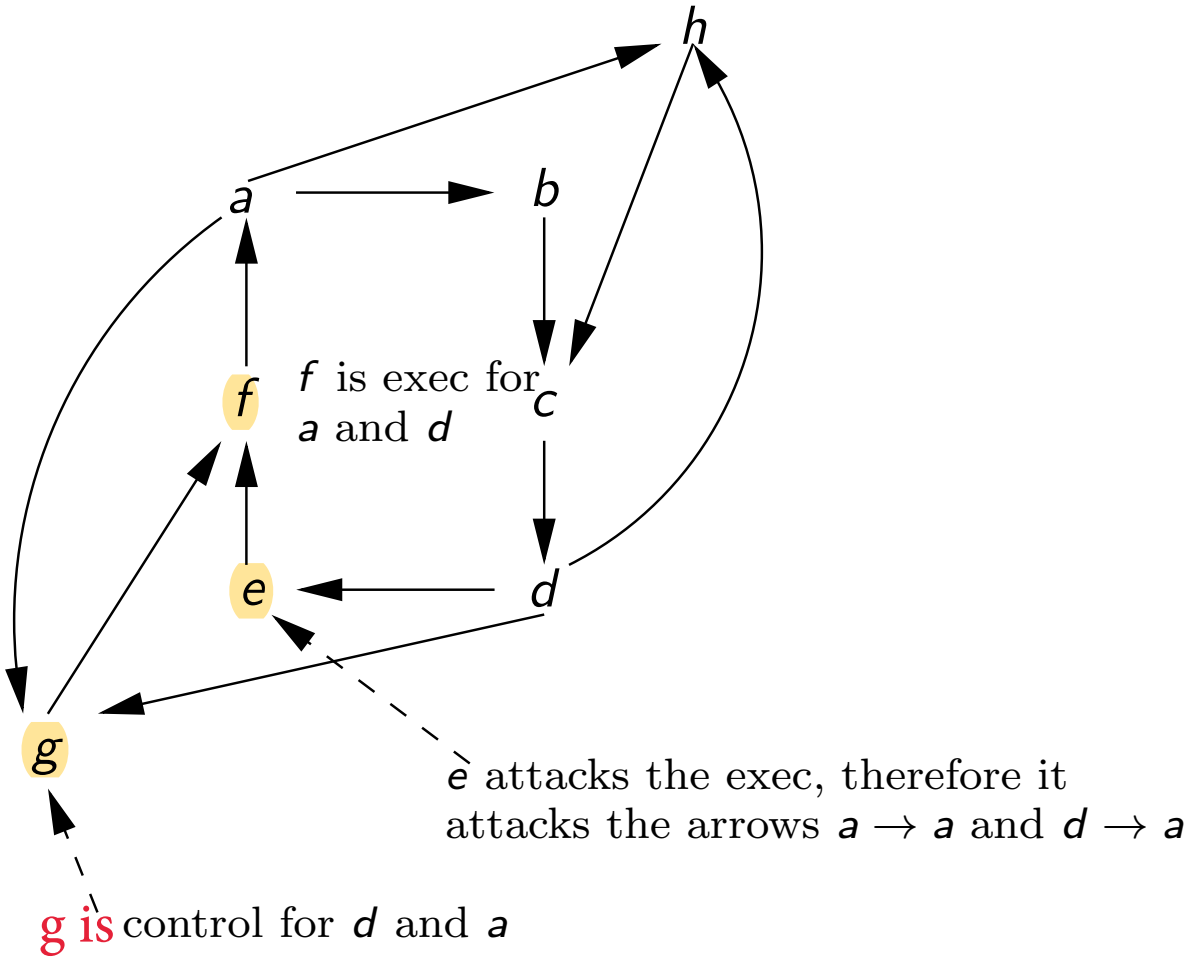


Figure: 27

Tim Example, different choice of control

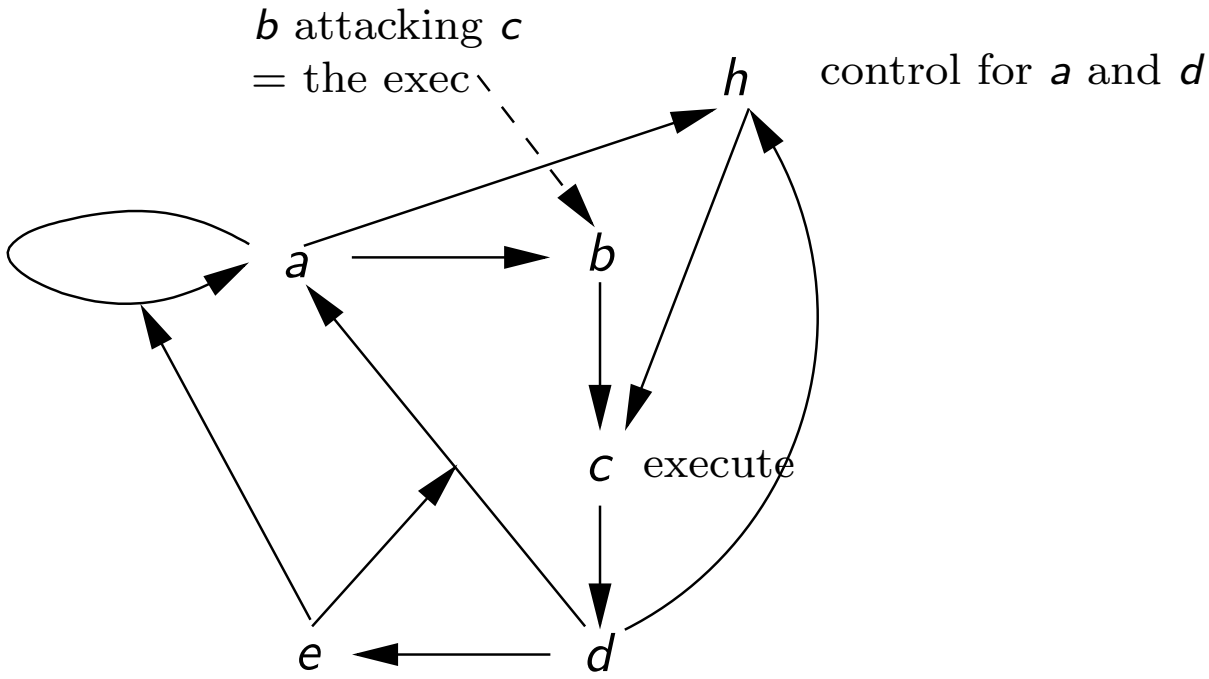


Figure: 28

So $d \to d$; new $a \to d$; and $b \to (d \to d), b \to (a \to d)$.

Tim Example, continued

Therefore we get Figure 29.

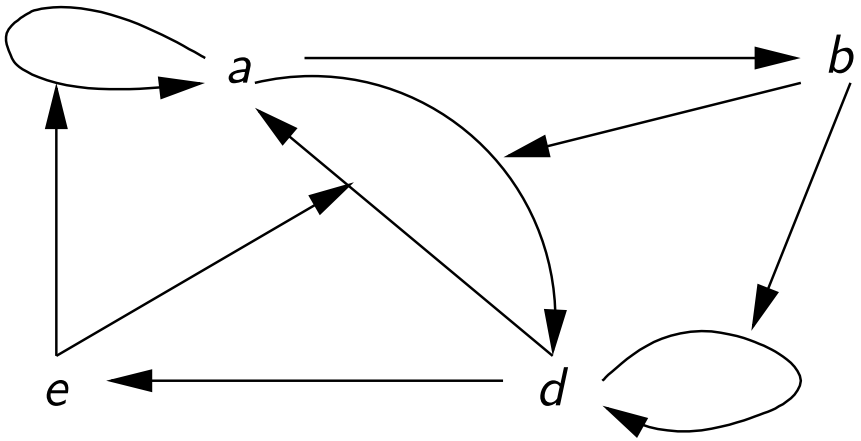


Figure: 29

Exercise: Guess extensions for Figure 29.

Reduction Principles

Let (S, R) be a full loop, an SCC.
Example E1

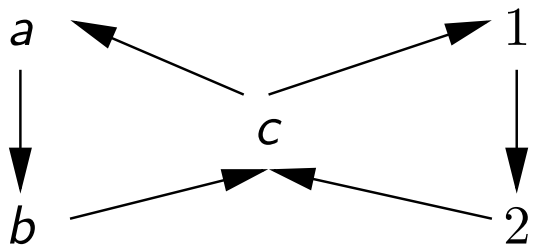


Figure: 30

Being a full SCC each point $C \in S$ must be attacked and must attack others.

Schematically,

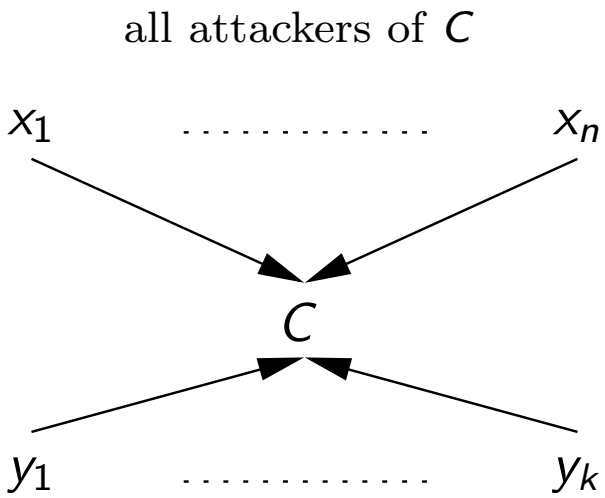
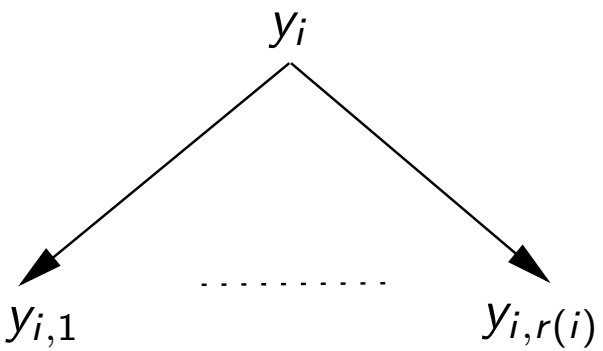


Figure: 31

Call C control for $x_1, \ldots, x_n$.
Call $y_1, \ldots, y_n$ execute for all and each of $x_1, \ldots, x_n$.

Let y_i attack $y_{i,1}, \ldots, y_{i,r(i)}$



$y_{i,j}$ are all points attacked by y_i

Figure: 32

Inductive action on (S, R)

Having chosen C and identified the attacks as in Figures 31 and 32 above.
Then:
Delete the control C and the execute points $y_1, \ldots, y_k$ as well as the arrows into and out of C, and connect new attacks from each $x_m, m = 1, \ldots, n$ onto each $y_{i,j}$.
We say the choice of C as control manages to reduce the number of points of the SCC.
Figure 33 illustrates what happens.
Figure 34 explains and shows that we may get new higher level attacks on the new arrows.

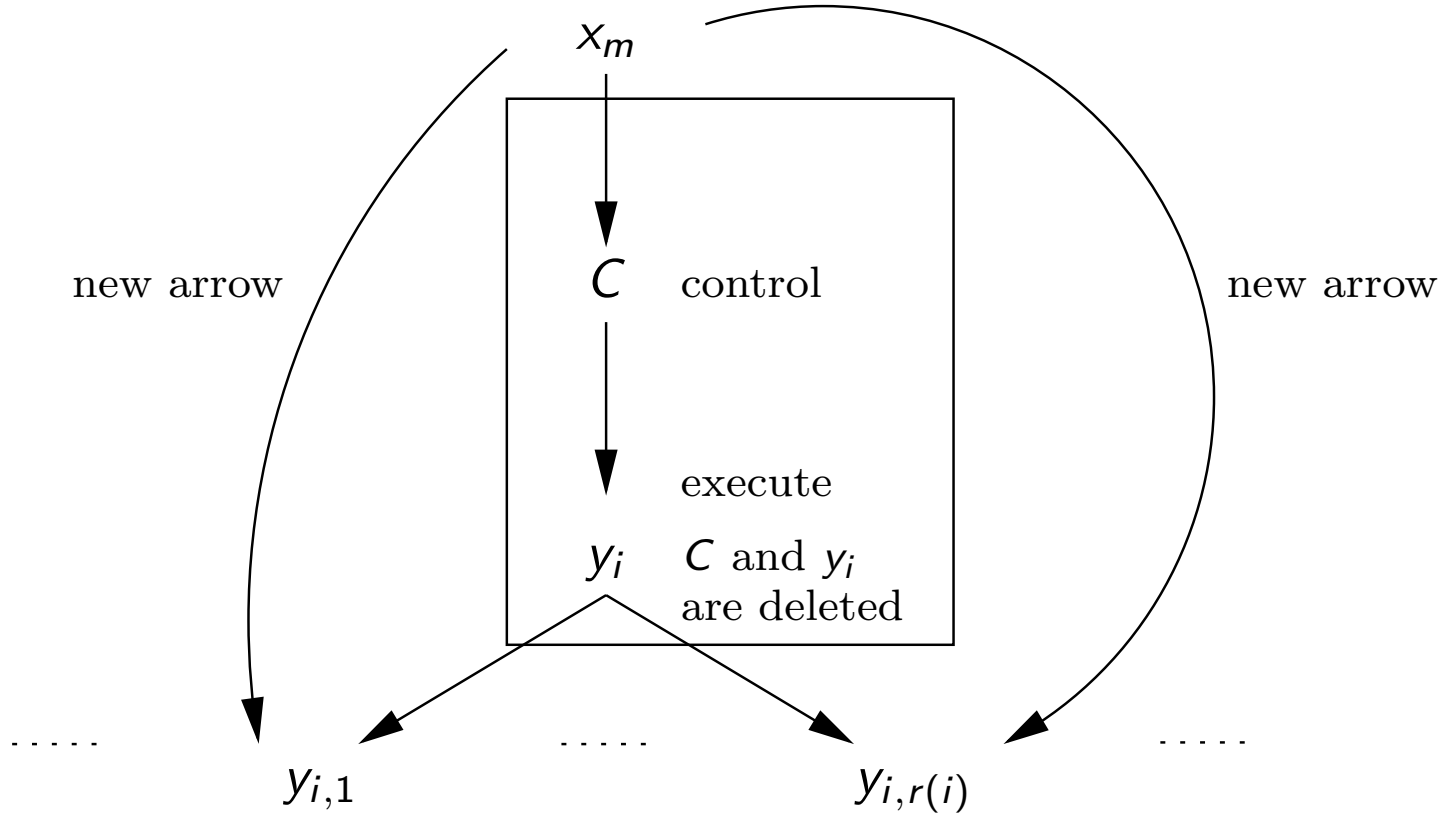


Figure: 33

Note: Any $z \to y_i$ becomes $z \to (x_m \to y_{i,j}), j = 1, \ldots, r(i)$.

z attacks the attack

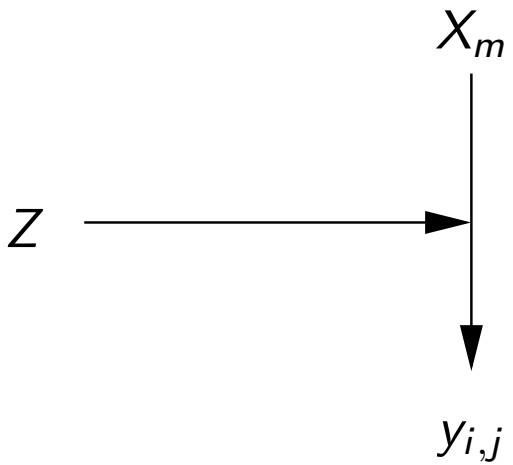


Figure: 34

Let us do this for Example E1 of Figure 30. We choose node C as control, we get Figure 35 below:

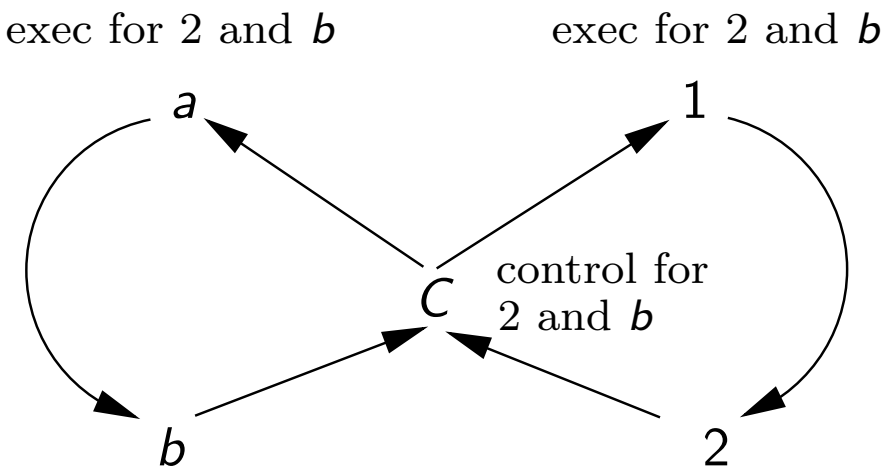


Figure: 35

We eliminate the service points and get Figure 36.

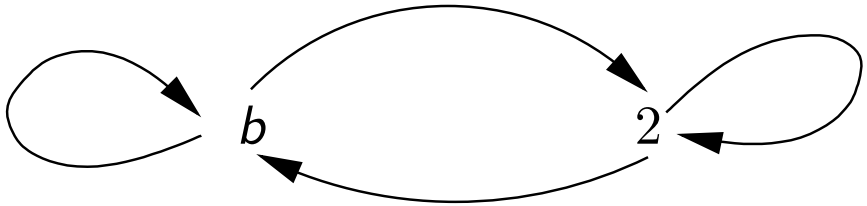


Figure: 36

If we now repeat the process to this new figure, we get the same figure. again

Summary of the process

Give (S, R) loop SCC

1. Choose C = control
2. Eliminate C and its executes
3. Whatever X_m attacks C, add direct attack to all targets of its executes
4. As in Figure 34, some z will attack the new attacks of 3.
5. Repeat.

Note that the number of nodes diminishes and the attacks from any $z \rightarrow$ execute points becomes higher level.
Repeat the process. After a while the process will stop.
Get extensions using another algorithm for higher level.
I have an algorithm for higher level!

Exercise X1

Apply procedure to Figure 37

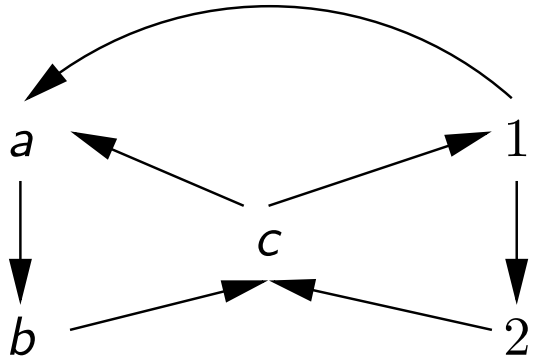


Figure: 37

You get

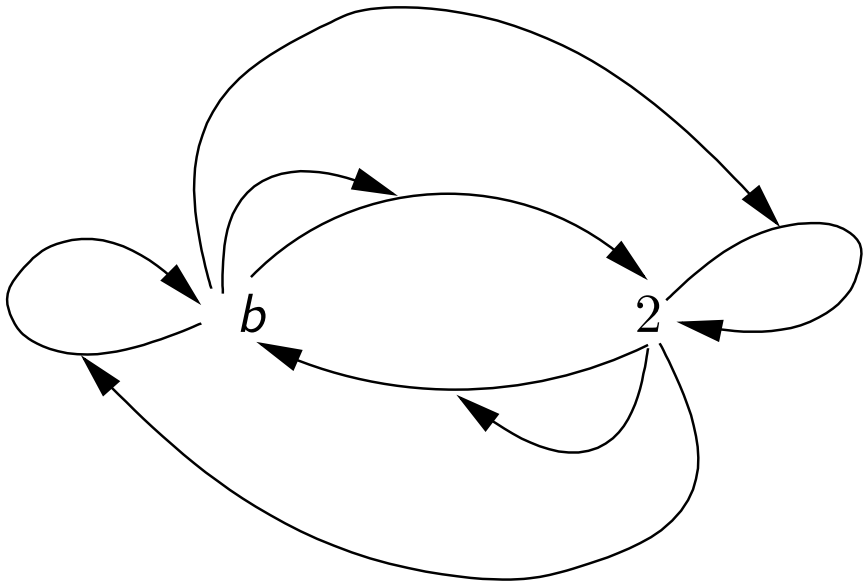


Figure: 38