The Handling of Loops in Talmudic Logic, with Application to Odd and Even Loops in Argumentation

Michael Abraham, Dov Gabbay and Uri J. Schild

Kings College

1 Background

The Talmud is a body of arguments and discussions about all aspects of the human agent's social, legal and religious life. It was completed over 1500 years ago and its argumentation and debates contain many logical principles and examples very much relevant to today's research in logic, artificial intelligence, law and argumentation.

In a series of books on Talmudic Logic, the authors have studied the logical prinicples involved in the Talmud, one by one, devoting a volume to each major principle

We have just finished writing Volume 5, entitled *Resolution of Conflicts and Normative Loops in the Talmud*, and the present paper describes how the Talmud deals with even and odd loops and compares the results with open issues in argumentation.

For other English papers corresponding to previous books, see [1, 2, 3, 4, 5, 6].

We start by looking at two typical loops, as in Figures 1 and 2.

$$\begin{tikzcd}a \arrow[r, bend left] & b \arrow[l, bend left]\end{tikzcd}$$

Figure 1:

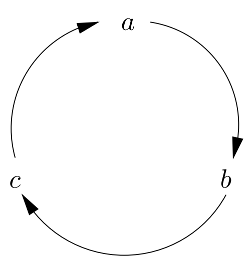

Figure 2:

We need to give some definitions.

An abstract network has the form (S, R), where S is a set of abstract nodes (arguments) and $R \subseteq S^2$ is the attack relation. Traditional research looks at extensions, these are subsets of S satisfying certain conditions (formulated in terms of R). Given (S, R) there may be several possible extensions of several types. In our case, for example, Figure 1 has three complete extensions $\{a\}, \{b\}$ and $\varnothing$, and Figure 2 has only one extension $\varnothing$.

Current research in argumentation, which relates to such loops and which connects with Talmudic logic, has two aspects:

1. Giving new definitions of extensions which can apply to abstract argumentation networks containing loops and allow us to get some new extensions other than "all undecided".

2. Adding extra information to the argumentation network which helps resolve the loops or help choose an extension.

The extra information one can add to the nodes of the network can be valuations or preferences among nodes. Mathematically one can look at valuations only, as preferences can be derived from them.

When we add valuations, we add a function $V : S \mapsto U$ where U is a value domain, giving some value to each $x \in S$.

V can be used in two extreme ways:

(a) Use V in the definition of extensions, by modifying the network or by disregarding and removing attacks, etc.

(b) Calculate the extensions without using V (i.e. ignoring V) and then using V to choose one's favourite extension or modify existing extensions and create new modified extensions.

(c) There is a third way, highly recommended by some members of the community, which is to use V in combination with the internal structure of the argument. (Note that V is not definded on arguments here but on components of arguments).

(a) is supported by Leila Amgoud and Trevor Bench-Capon.

(b) is supported by the 1500 years old Talmudic logic and recently by a 2010 paper by Toshiko Wakaki, [11].

(c) is supported by Henry Prakken in a 2010 paper [9].

The (b) and (c) approaches maintain consistency while (a) is problematic. See a critique by Martin Caminada [12]. We are grateful to Martin Caminada for providing us with the above information, as well as sending us his critique of aproach (a).

Our plan for this paper is very simple. In Section 2 we present the notion of Shkop extension to an abstract network (S, R) and compare it with Baroni's and his colleagues [13, 15] CF2 extensions.[1]

In Section 3 we discuss some counter examples by Martin Caminada. In Section 4 we conclude the paper. In a follow-up paper, yet to be written, we give examples of how the Talmud offers valuations to resolve loops of odd and even types and how the Talmud chooses extensions.

2 Shkop extensions

We begin with a motivating Talmudic example, the dates are all in the same year, say 2010.

Example 2.1 (The divorce). *Jane is married to John. She develops some feelings for Frank and wants a divorce from John. Frank is a rich man and promises to compensate John generously if he cooperates. We now have the following temporal sequence:*

[1]Rabbi Shimon Shkop, 1860–1930. A Talmudic scholar analysing many logical principles in the Talmud.

Jan 01: *John gives divorce papers to Jane. The divorce is conditional on Jane marrying Frank by the 31st of March. Such conditional divorces are allowed in the Talmud. If Jane marries Frank before 31st March then all is well. If Jane does not marry Frank by 31st March then the divorce papers, the beginning, from January 01 are nullified and the divorce is not valid from Jan 01. This is Talmudic legal backwards causality.*

Feb 01: *Jane takes her divorce papers and marries Terry. This marriage is valid because Jane's divorce papers are valid. Jane can still potentially fulfill the condition mentioned in the divorce papers; she can still divorce Tery and marry Frank.*

31st March: *Jane, without getting a divorce from Terry, goes and marries Frank.*

There is no doubt that Jane is a naughty girl! Frank is a bit paranoid, asking John to give Jane a conditional divorce.

Now we seem to have landed in a logical loop.

Let us build up an argumentation network based on this story.

The base logic is classical temporal logic. The base theory in the logic is the following:

1. *If x is married to someone then x cannot marry someone else.*
2. *If x is married to y at time t then x continues to be married to y until there is a divorce or death.*
3. (a) *A divorce can be given at time t, conditional on an action taken at time $s > t$.*
   (b) *If the action is not taken at time s then there is backward causality and the divorce is not valid from time t.*
   (c) *If the action is taken at time s then the divorce is valid at time t.*
   (d) *At any time $t', t \leq t' < s$, the divorce given at time t on a condition to be fulfilled at time s, is considered valid at time t' as long as there is the reasonable possibility, as seen from time t', that the condition will be fulfilled at time s.*
4. **Fact***: John gave a divorce to Jane on January 01, conditional on Jane marrying Frank by March 31.*
5. **Fact***: Jane married Terry on Feb 01.*
6. **Fact***: Jane married Frank on March 31, without ever getting a divorce from Terry.*
7. *Note: It is possible for x to give a divorce to y at time t on the condition that y marries z $(z \neq x)$ at time $s > t$.*

   One might argue that at time s, we have a problem:

   y is still married to x therefore y cannot marry z. It is only when y marries z at time s that y is divorced from x at time t and is therefore able to marry z at time s.

   Since we allow for such conditions, we regard marrying z at s and enabling the divorce at t as simultaneous.

   The answer is that the condition of marrying z is not an enabling condition for the divorce papers but a nullifying condition. If it is not fulfilled the divorce papers are nullified.

We now consider the following arguments seen from the temporal point of view of March 31.

DJJ- *John's divorce from Jane on January 01 is not valid.*
The reasoning in this argument from base data goes as follows:

On February 01, the divorce was valid because there was the possibility of fulfilling the condition of the divorce, from Rule (3d). Therefore the marriage to Terry (Fact (5)) is valid and does not nullify the divorce, since Jane can still divorce Terry and marry Frank (Rule (3d)).

Therefore at the time March 31, when Jane married Frank without divorcing Terry, her marriage to Frank was not valid (Rules (1) and (2)). Hence, since the condition of the divorce was not fulfilled, the divorce is not valid.

MJT+ *Jane's marriage to Terry on Feb 01 is valid.*
The argument goes as follows:

Since Jane got a conditional divorce from John and the condition can still be fulfilled her divorce stands and she can marry Terry.

MJF+ *Jane's marriage to Frank is valid.*
the argument for that is as follows:

Assume the marriage to Frank is not valid. Then Jane's divorce from John is not valid. Hence her marriage to Terry is not valid. But then Jane has a conditional divorce from John and she is not married to Terry, therefore she is free to marry Frank and the marriage is valid. Therefore since $\neg$**MJF+** $\to$ **MJF+**, *we conclude* **MJF+**.

We now get the argumentation loop presented in Figure 3.

$$\begin{tikzcd}
\mathbf{MJF+} \arrow[dr] & & \mathbf{MJT+} \arrow[ll] \\
& \mathbf{DJJ-} \arrow[ur] &
\end{tikzcd}$$

Figure 3:

It is clear that we have an odd loop here and the only Dung extension is $\varnothing$, being all undecided. However, life must go on and we need a resolution as to whom Jane is married to! Is she married to John, to Terry or to Frank?!

Here we introduce the intuitive Rabbi Shkop principle:

Shkop principle

If by assuming $x =$ in, we deduce that $x =$ out, then surely x must be out.

Let us apply this to our example. We have three possibilities for the choice of x, see Figure 3.

1. $x =$ **DJJ**$-$
2. $x =$ **MJT**$+$
3. $x =$ **MJF**$+$

We reason against the direction of the attack arrows. This reasoning is done later on, see Example 2.6 below for the calculation.

We get three extensions for each one of the choices of x:

1. Marriage to Frank is valid.

2. Divorce not valid — Jane is married to John.

3. Marriage to Terry is valid.

Commonsense dictates that we should not test the validity of the divorce because at the time (and here we make use of the temporal sequence) we did not know what Jane was going to do. Similarly we should not test the validity of the marriage to Terry because Jane could still have divorced him. So the only test is that of the validity of marriage to Frank. This test gives by the Shkop principle that **MJF+** = out and therefore the network looks like Figure 4, (see also Example 2.6 below for a detailed analysis).

a is an annilhilator node

making sure **MJF+** is out

$$\begin{tikzcd}
a \arrow[d] & & \\
\mathbf{MJF+} \arrow[dr] & & \mathbf{MJT+} \arrow[ll] \\
& \mathbf{DJJ\text{-}} \arrow[ur] &
\end{tikzcd}$$

Figure 4:

Figure 4 has the extension:

$$\begin{array}{lcl}
a & = & \text{in} \\
\mathbf{DJJ}- & = & \text{in} \\
\mathbf{MJT}+ & = & \text{out} \\
\mathbf{MJF}+ & = & \text{out}
\end{array}$$

In the above considerations we kept the temporal aspects in the metalevel. We can include these aspects in the object level. We time stamp each argument and each attack arrow, according to the way the story unfolds. If we do this we get Figure 5.

Obviously the loop occurs on March 31. So we have to do the Shkop test on the March 31 argument, which is **MJF+**.

In general we can talk about Shkop temporal argumentation frames of the form $\mathbf{N} = (S, R, \mathbf{T})$, where (S, R) is an ordinary network and $\mathbf{T}$ is a time stamping function:

$$\mathbf{T} : S \cup R \mapsto \text{ Time axis.}$$

For any choice of time t we look at the network

$$\mathbf{N}_t = (S_t, R_t),$$

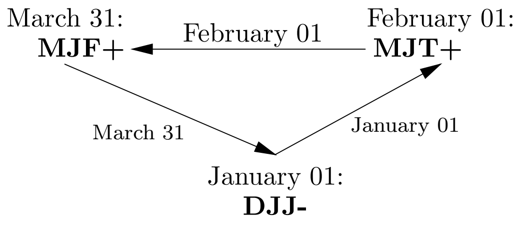

Figure 5:

where

$$S_t = \{a \in S | \mathbf{T}(a) \leq t\}$$
$$R_t = \{(x, y) \in R | \mathbf{T}(x, y) \leq t\}$$

Given $a \in S$ with $\mathbf{T}(a) = s$, we check according to Shkop the test $a = 1$? in the network $\mathbf{N}_s = (S_s, R_s)$.

Let us now be a bit more formal about Shkop extensions. Our aim is to offer the argumentation community the notion of Shkop semantics, and compare it with CF2 or Stage semantics. To do that, we need to generalise the intuitive Shkop principle in a sensible way.

For reasons of clear exposition, we find it advantageous to actually start from a recent paper of Martin Caminada, entitled Preferred semantics as Socratic discussion [10].

Caminada sets himself to give a game theoretic answer to the question:

Q: Given (S, R) and $a \in S$, can a be an element of some admissible extension?

His method is to assume that $a =$ in and see by Socratic discussion whether such a position can be maintained. The method is best explained by two examples.[2]

Example 2.2. *Consider Figure 6*

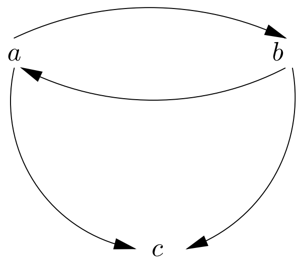

Figure 6:

We ask can we have $c =$ in in some extension? We proceed as follows:

1. *$c =$ in, assumption*
2. *$b =$ out, from (1)*
3. *$a =$ in, from (2)*

[2]Appendix A offers a Tableaux algorithm for this test.

4. *$a =$ out, from (1)*

We get a contradiction. The assumption $c =$ in, lead us, using the attack rules and the geometry of the figure, that both $a =$ out, and $a =$ in.

Thus the answer the question about c is that it cannot be in, it must be out.

Example 2.3. *Consider Figure 2. Ask the question can $c =$ in? Let us check:*

1. *$c =$ in, assumption*
2. *$b =$ out, from (1)*
3. *$a =$ in, from (2)*
4. *$c =$ out, from (3)*

So again the answer is negative, there is no extension in which $c =$ in.

Remark 2.4. *Note that the proofs in the Caminada Socratic discussion obtain a contradiction by using the direction in the graph against the arrow. Thus if we have*

$$x \to y \to z$$

and we assume $y =$ in, Caminada is allowed to deduce $x =$ out, going against the arrow, but is not allowed to deduce $z =$ out going with the arrow.

It seems that even with this restriction, the Socratic discussion is strong enough to identify all nodes a in the network for which $a =$ in is impossible.

Caminada's paper stops when we get our answers to the question of whether $a =$ in is possible or not.

Now let us use these two examples to explain what Shkop does. Shkop introduced a principle for resolving loops:

Shkop's original principle
If the test assumption $a =$ in leads to the conclusion that $a =$ out, then a must be annihilated and be out.

To implement such a principle we need some notation. Let (S, R) be a network and let the elements of S be denoted by lower case letters. Let us add for any $a \in S$ a new annihilator letter, capital A.

With the above notation, let us redo Examples 2.2 and 2.3 according to Shkop.

Example 2.5 (Doing Example 2.2 according to Shkop)**.** *We start by testing $c =$ in in Figure 6.*

1. *$c =$ in, test assumption*
2. *$b =$ out, from (1)*
3. *$a =$ in, from (2)*
   The Caminada Socratic discussion goes against the arrow and would continue

4*. *$a =$ out, from (1), a contradiction, because we get both $a =$ in and $a =$ out.*
The Shkop original principle requires us to get $c =$ out for a contradiction, because our original test was for $c =$ in?. Therefore we need to go forward with the arrow using (3), as this is the only way to get back to c, and get (4) below. Going forward:

4. *$c = out$, from (3)*

5. *From (1)–(4) we get that c must be annihilated by the Shkop principle.*

This means that we replace Figure 6 by Figure 7.

We may now feel comfortable, allowing ourselves to go both backwards and forwards with the arrow, and thus maintaining the intuitive spirit of the Shkop principle. This, however, is problematic. Caminada has shown a counter example which is problematic. We discuss this later in Section 3. So we cannot allow ourselves to prove forward with the arrow. So we need to modify the Shkop principle.

Our choice of modifying the Shkop principle is to state:

- *If $a = in$ leads to a contradiction then a must be out. In deriving the contradiction, we use reasoning going backwards with the arrow only, see Appendix A. Once the contradiction is derived we introduce an annihilator for a.*

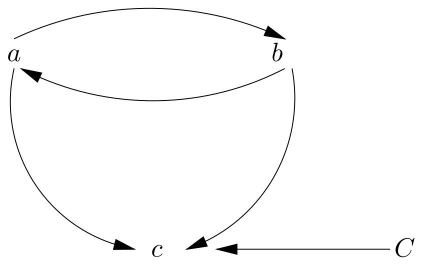
$$\begin{tikzcd}
a \arrow[rr, bend left] \arrow[dr, bend right] & & b \arrow[ll, bend left] \arrow[dl, bend left] & \\
& c & & C \arrow[ll]
\end{tikzcd}$$

Figure 7:

Coming back to the argumemtation network of Figure 6, having tested $c = in?$, we can continue to test $a = in$ and test $b = in$ but this will not require any more annihilators.

The Shkop extensions for Figure 6 are obtained by taking ordinary extensions for Figure 7 and ignoring the annihilators. In the case of Figure 6 the Shkop procedure made no difference but for Figure 2 it does as it resolves loops.

Example 2.6 (Doing Example 2.3 according to Shkop)**.** *We have three tests to conduct:*

Test 1: $c = in$

Test 2: $b = in$

Test 3: $a = in$

Test 1

1. *$c = in$, test assumption*

2. *$b = out$, from (1)*

3. *$a = in$, from (2)*

4. *$c = out$, from (3)*

5. *Using the Shkop principle c must be annihilated and Figure 2 replaced by Figure 8.*

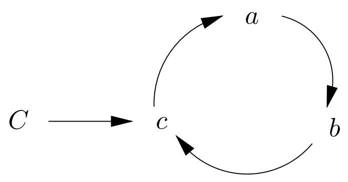
$$\begin{tikzcd} & a \arrow[dr, bend left] & \\ C \arrow[r] & c \arrow[u, bend left] & b \arrow[l, bend left] \end{tikzcd}$$

Figure 8:

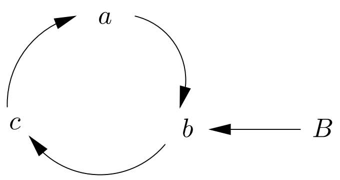
$$\begin{tikzcd} & a \arrow[dr, bend left] & \\ c \arrow[ur, bend left] & & b \arrow[ll, bend left] & B \arrow[l] \end{tikzcd}$$

Figure 9:

Figure 8 is a new network and we can apply the Shkop test to it. We will get no more contradictions. The Dung extension for it is $\{C, a\}$.

The other tests will give us Figures 9 and 10.

The normal Dung extensions for Figures 9 and 10 are respectively $\{B, c\}$ and $\{A, b\}$.

According to Shkop, the Shkop extensions for Figure 2 are obtained from the normal extensions of Figures 8, 9 and 10 by ignoring the annihilators letters.

Thus we get the extensions $\{a\}, \{b\}, \{c\}$. Notice that these are the conflict free sets of Figure 2.

We ask the reader to remember this because we shall compare the Shkop extensions with Baroni's CF2 extensions.

Remark 2.7. *The reader should note that the Shkop procedure was originally intended for elements x of a network which are part of an odd loop, see Example 2.1. Once the element is found to be out by the Shkop principle, we move to a new network containing the annihilator X of x and we deal with the new network only. Shkop would never test $c =$ in ? immediately (at that moment, if we take into account the temporal aspect, see Section 4) in Figure 1 because c is not part of a loop. He would test $a =$ in? and $b =$ in? and find no contradiction. For the sake of mathematical completeness and generalising Shkop, we can allow the use of the Shkop principle to any x in the network. The test is similar to the Caminada Socratic discussion (see Appendix A for a Tableaux algorithm doing the same as Caminada's Socratic discussion), and if $x =$ in is found contradictory, this means that x must be out. Thus adding the annihilator X with $X \to x$ to the network will make no difference and we get an equivalent network.*

We therefore put forward the Generalised Shkop Principle:

Generalised Shkop Principle

Let (S, R) be a network and let $a \in S$. If the assumption $a =$ in leads to a contradiction (i.e. for some $x \in S$, we get both $x =$ in, and $x =$ out) by reasoning only backward against the direction of the arrow (as Caminada does in his Socratic discussion, or as we do in Appendix A using Tableaux) then a must be out. To ensure that a is out, we move to a new network $(S \cup \{A\}, R \cup \{A, a)\})$, where A is a new letter, being the annihilator of a.

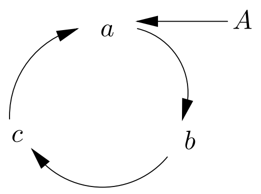
$$\begin{tikzcd} & a \arrow[dr, bend left] & & A \arrow[ll] \\ c \arrow[ur, bend left] & & b \arrow[ll, bend left] & \end{tikzcd}$$

Figure 10:

Note that Caminada proved in his Socratic paper that for a network (S, R) and $a \in S$ the condition:

- The assumption $a =$ in leads to a contradiction by correctly reasoning backwards against the direction of the arrow.

is equivalent to the declarative condition

- a is not a member of any admissible set.

We can therefore formulate the Generalised Shkop principle in an equivalent declarative way as follows:

Generalised Shkop principle (declarative)

Let (S, R) be a network and let $a \in S$. If a is not a member of any admissible set, then a must be out. To ensure that a is visibly out we move to a new network $(S \cup \{A\}, R \cup \{(A, a)\})$, where A is a new letter, being the annihilator of a.[3]

We are now ready to define the notion of Shkop extensions.

Definition 2.8. *1. Let $\mathbf{N} = (S, R)$ be a finite argumentation network. Assume elements $y \in S$ are denoted by lower case letters. For each such y let Y be the annihilator of y.*

We define by induction the notions of

(a) $(y_1, \ldots, y_k), y_i \in S$ is a legitimate Shkop sequence.

(b) $\mathbf{N}_{(y_1,\ldots,y_k)}$ is a Shkop model dependent on $(y_1, \ldots, y_k)$.

Case $k = 1$

y_1 is a legitimate Shkop sequence if y is not a member of any admissible set of (S, R) (or equivalently by Caminada [10], if the assumption $y =$ in, in (S, R) leads to a contradiction using Caminada Socratic discussion). In this case let

$$\begin{aligned} \mathbf{N}_{y_1} &= (S \cup \{Y_1\}, R \cup \{(Y_1, y_1)\}) \\ &= (S_{y_1}, R_{y_1}). \end{aligned}$$

Case $k + 1$

Assume $(y_1, \ldots, y_k)$ is a legitimate sequence and assume $\mathbf{N}_{(y_1,\ldots,y_k)}$ is well defined. Let $y_{k+1} \in S$ be a point such that y_{k+1} is different from all $y_1, \ldots, y_k$. Assume that y_{k+1} is not a member of any admissible set in $\mathbf{N}_{(y_1,\ldots,y_k)}$, (or equivalently the assumption $y_{k+1} =$ in, in the network $\mathbf{N}_{(y_1,\ldots,y_k)}$ leads to a contradiction using Caminada Socratic discussion).

[3]So for example the Liar paradox network $(\{a\}, \{a \to a\})$ becomes the network $(\{A, a\}, \{a \to a, A \to a\})$.

Then $(y_1, \dots, y_{k+1})$ is a legitimate sequence. Let $\mathbf{N}_{(y_1,\dots,y_{k+1})}$ be $(S_{(y_1,\dots,y_{k+1})}, R_{(y_1,\dots,y_{k+1})}$, where

$$\begin{array}{rcl} S_{(y_1,\dots,y_{k+1})} & = & S_{(y_1,\dots,y_k)} \cup \{Y_{k+1}\} \\ R_{(y_1,\dots,y_{k+1})} & = & R_{(y_1,\dots,y_k)} \cup \{(Y_{k+1}, y_{k+1})\}. \end{array}$$

2. *Let $(y_1, \dots, y_k)$ be a legitimate sequence. Let n be the number of elements of S. Then we say the rank of $\mathbf{N}_{(y_1,\dots,y_k)}$ is $n - k$.*

3. *Let $(y_1, \dots, y_k)$ be a legitimate sequence. Let $\mathbf{N}_{(y_1,\dots,y_k)}$ e its associated Shkop network. We say $\mathbf{N}_{(y_1,\dots,y_k)}$ or equally $(y_1, \dots, y_k)$ is* clean *iff there are no legitimate sequences extending $(y_1, \dots, y_k)$. Alternatively, iff for any $y \in S, y \neq y_i, i = 1, \dots, k$, we have that the test $y = in$ does* not *lead to a contradiction.*

4. *Let $\mathbf{N}_{(y_1,\dots,y_k)}$ be clean. Then we define the set of Shkop extensions of $\mathbf{N} = (S, R)$ as derived from $(y_1, \dots, y_k)$.*
   Notation
   $$\mathbb{E}^{\text{Shkop}}_{(y_1,\dots,y_n)}$$
   to be defined as follows.

   Let E be any ordinary Dung extension of $\mathbf{N}_{(y_1,\dots,y_k)}$ or equivalently let λ be any Caminada labelling for $\mathbf{N}_{(y_1,\dots,y_k)}$, then $E \cap S$ (or equivalently) $\lambda \upharpoonright S$ be an element of $\mathbb{E}^{\text{Shkop}}_{(y_1,\dots,y_k)}$.

5. *We now define the notion of all Shkop extensions of a finite network $\mathbf{N} = (S, R)$. We define the set of all Shkop extensions of $\mathbf{N}$ to be*
   $$\mathbb{E}^{\text{Shkop}}_{\mathbf{N}} = \bigcup_{\substack{(y_1,\dots,y_k) \\ \text{clean}}} \mathbb{E}^{\text{Shkop}}_{y_1,\dots,y_k)}$$

Remark 2.9. *Note that this is our definition based on the generalised Shkop principle. We can give restricted variations of it. For example, following Baroni* et al. *in their paper [15] of SCC recursiveness, we can first rewrite (S, R) as an acyclic ordering of maximal loops and then apply the Shkop procedure to loop elements starting from the top loops. This is like the way the CF2 extensions are calculated. We shall give a substantial example below to show you what happens.*

It is now time to give some more Shkop examples.

Example 2.10. *Consider Figure 11*

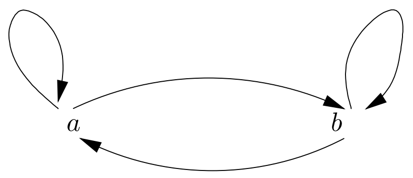

Figure 11:

Testing b and then testing a or testing a and then testing b will lead to the same Figure 12. Therefore the Shkop extension of Figure 11 is $\{a = out, b = out\}$.
This does not contradict the usual Dung extension of all undecided!

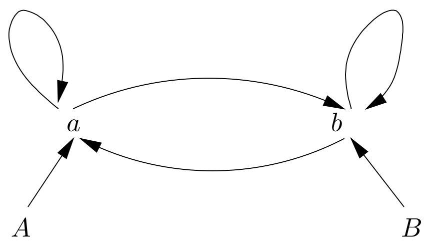

Figure 12:

Example 2.11 (Shkop compared with CF2)**.** *Consider the network in Figure 13. This figure appears in [8] as an example of how Baroni's CF2 semantics works. Gaggl and Woltran have a program which can compute the CF2 extensions.*

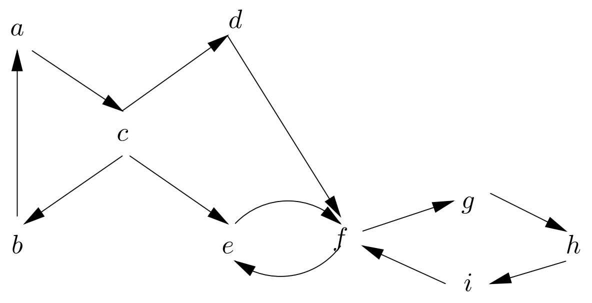

Figure 13:

The CF2 extensions for Figure 13 are the following:

$$\begin{aligned} E_1 &= \{c, f, h\} \\ E_2 &= \{c, g, i\} \\ E_3 &= \{b, d, e, g, i\} \\ E_4 &= \{a, d, e, g, i\}. \end{aligned}$$

The CF2 semantics would start with the top cycle $\{a, b, c\}$. They would take maximal conflict free subsets which are in this case $\{a\}, \{b\}, \{c\}$ and then arbitrarily decide on the three assignment:

1. *$c =$ in, $b =$ out, $a =$ out*
2. *$b =$ in, $c =$ out, $a =$ out*
3. *$a =$ in, $c =$ out, $a =$ out*

Having now given values to a, b and c, one can propagate the values to the rest of the network and get extensions.

For example:

If $c =$ in, then $d = e =$ out.

Therefore $f =$ in and hence $g =$ out, $h =$ in and $i =$ out.

We got ourselves an extension by breaking the loop $\{a, b, c\}$. The alternative, if we follow traditional Dung style approach is to have one extension only = all undecided.

The method seems arbitrary, a technical device to generate extensions. It does not always help. Consider Figure 14 for example.

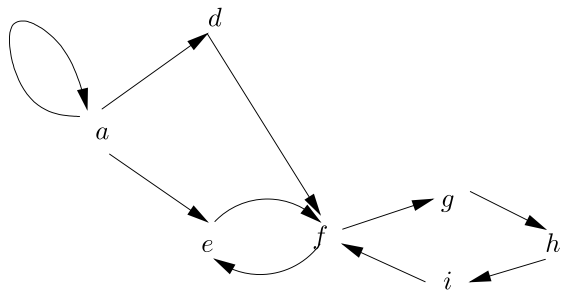

Figure 14:

Now let us look at Shkop extensions of Figure 13.

Option 1
Accept the procedure where we start from the top loops. Call this top-down Shkop procedure. In this case we start from $\{a, b, c\}$ *and ask, as in Example 2.6,*

Test 1: $a = in$
Test 2: $b = in$
Test 3: $c = in$

This will yield Shkop figures 15, 16 and 17.

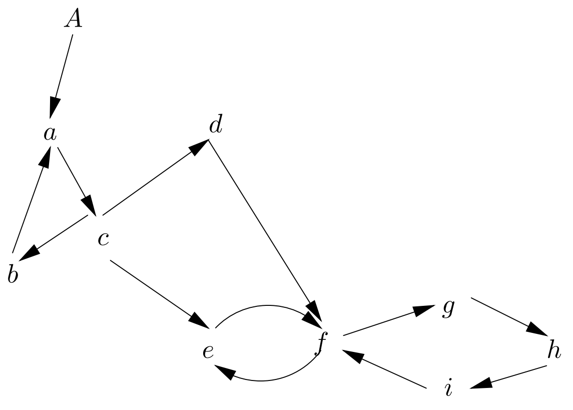

Figure 15:

From Figure 15 we get the extensions E_1 *and* E_2*. From Figure 16 we get Extension* E_4 *and from Figure 17 we get extension* E_3*.*

In the case of Figure 14, using the Shkop procedure on $a = in$ *will give Figure 18.*

and we get the extension $\{d, e, g, i\}$*.*

Let us now check what happens if we allow the Shkop process to start from any point. Let us start with $d = in?$ *and then* $e = in?$*. We will get that both need to be annihalated. If we*

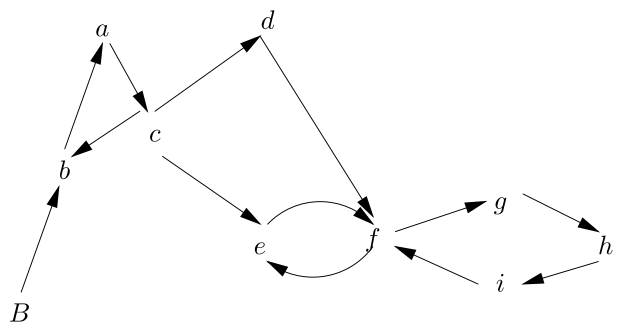

Figure 16:

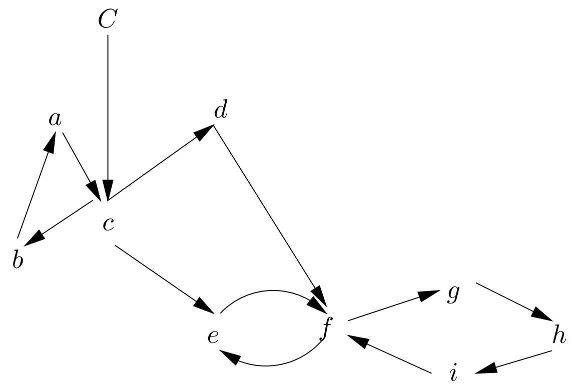

Figure 17:

carry on asking $a =$ in? or $b =$ in? or $c =$ in? we get the extensions

$$\{c, f, h\}$$
$$\{c, g, i\}$$
$$\{a, f, h\}$$
$$\{a, g, i\}$$
$$\{b, f, h\}$$
$$\{b, g, i\}$$

This paper is mainly qualitative. A more mathematical exposition will need to address some open problems.

Problem 1
Under what circumstances is the top down Shkop process the same as CF2?

Problem 2
Is there a set of equations in the equational approach [7] characterising say the top down Shkop extensions?

Note that all extensions obtained by the Shkop procedure are stable. There is no undecided. Shkop kills all undecided!

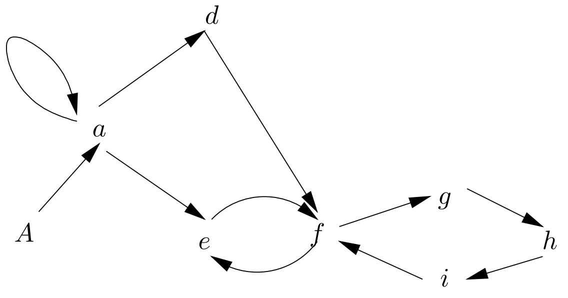

Figure 18:

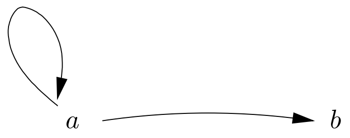

Figure 19:

Remark 2.12. *The reader may seek some meaning to the Shkop algorithm. For this, see the conclusion Section 4. The reader must remember that Talmudic logical argumentation and debate was conducted from the first to the end of the fifth centuries and was used in Jewish communities in the world during the following 1500 years.*

Rabbi Shkop just explained the principles involved and we in this paper are formally modelling them in terms of known abstract argumentation methods.

The principle works!

Remark 2.13 (Comparison with stage semantics)**.** *Stage semantics is discussed in detail in [17]. It has similarities with the Shkop extensions but it is not the same. Both ignore self loops but stage semantics may ignore arguments which are not attacked by any other argument. Shkop extensions never do that.*

We take our examples from [17]. Consider Figure 19

Stage semantics will ignore the self looping a and will have the extension $\{b\}$*. The same is the case with the Shkop semantics. They both agree on* $b = in$*. Stage will say* $a = undecided$ *while Shkop will say* $a = out$*.*

As a second example from [17], take Figure 20.[4] *Shkop will not agree here with the traditional extension. According to Shkop we have*

$$a = in,\ b = out,\ c = out.$$

The traditional extension will have

$$a = in,\ b = out,\ c = undecided.$$

Stage semantics allows for two extensions: the first one is the same as the traditional one

$$a = in,\ b = out,\ c = undecided.$$

[4]In fact Pietro Baroni and Massimilano Giacomin invented this figure in order to show that CF2 semantics has some advantages above stage semantics.

The second one is

$a =$ undecided, $b =$ in, $c =$ out.

Note that in the second stage a is not in, even though it has no attackers. This is rather strange. Caminada has proved, however, that every argumentation network has at least one stage extension which contains its ground extension. So it can be well behaved. Compare the stage semantics result for the network of Figure 20 with Example 3.1 and the considerations leading to Figure 21. We get the stage semantics if we go forward. Is this a coincidece? We think it is.

3 Caminada counter examples: A discussion

Martin Caminada read an earlier version of Section 2 and gave us penetrating comments and devastating counter examples. The aim of this Section is to put forward an alternative formulation of the Shkop principle which maintains the spirit of Shkop while avoiding the counter examples of Caminada.

We need to summarise the intellectual chain of reasoning events.

(1) The original Shkop principle, as formulated by Shkop, says as follows:

(*1) Let $\mathbf{N} = (S, R)$ be a network. Let $x \in S$. Assume (test) $x =$ in. If one can prove that this entails $x =$ out, then surely x must be out.

Our modelling of this principle was to move to the network $\mathbf{N}_x$, as defined in Definition 2.8.

Shkop does not specify what it means "to be able to prove that $x =$ in entails $x =$ out". We adopted the Caminada Socratic method to give meaning to this notion.

(2) Here we had a problem. Caminada's method uses reasoning against the direction of the arrow. So if we have, for example

$$y \to x \to z$$

and we test the assumption $x = 1$, then Caminada allows us to deduce $y = 0$, but we are not allowed to deduce $z = 0$.

The difficulty with this is that Shkop formulated his principle by saying "$x =$ in can prove $x =$ out".

It is the same x.

The "same x" restriction is OK for cases of pure loops of the form

$$x \to a_1 \to a_2 \to \ldots \to a_k \to x$$

We can prove $x =$ in implies $x =$ out by going backwards, but for cases like Figure 6 (the test case assuming $c =$ in) we cannot get $c =$ out by going against the arrow only, as discussed in Example 2.5.

Our original modification of Shkop principle was to allow forward reasoning with the arrow. However, Caminada landed a devastating counter example on this attempt (see Example 3.1 below).

We therefore reformulated the generalised Shkop principle in a safe way, as follows.

(*2) Let $\mathbf{N} = (S, R)$ be a network. Let $x \in S$. Assume (test) $x =$ in. If one can prove a contradiction from this assumption, say that for some $y \in S$, both $y =$ in and $y =$ out are derivable, then surely x must be out, and move to $\mathbf{N}_x$

The above is equivalent to the following (in view of Caminada's Socratic paper).

(*3) Let $\mathbf{N} = (S, R)$ and let $x \in S$. If x is not part of any extension (equivalently if there is no Caminada labelling λ with $\lambda(x) =$ in), then surely x must be out and we move to $\mathbf{N}_x$

So the Shkop extensions and Shkop semantics are obtained by systematically annihilating all points which cannot be part of an extension, as defined in Definition 2.8. This is a Draconian instrument. Note that it needs to be done in sequence, one node at a time.

Example 3.1 (Caminada's counter example)**.** *Consider Figure 20*

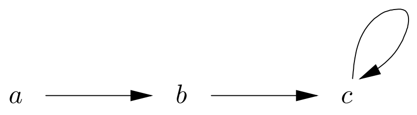
$$\begin{tikzcd} a \arrow[r] & b \arrow[r] & c \arrow[loop above] \end{tikzcd}$$

Figure 20:

Let us test $a = 1$? allowing reasoning both with and against the arrow. We reproduce Caminada's reasoning

1. $a =$ in, assumption
2. $b =$ out, from (1)
3. We now do case analysis for c.

   Case 3a $c =$ in
   In this case we continue

   (4a) $c =$ out, since c attacks itself.
   (5a) $b =$ in, from (4a)
   (6a) $a =$ out, from (5a)

   Case 3b $c =$ out

   (4b) $b =$ in, since $c =$ out
   (5b) $a =$ out, from (4b)

4. Since in both cases we get $a =$ out, then by the Shkop principle surely $a =$ out and we move to Figure 21.

   Clearly this is not acceptable.

Later on we shall modify the forward proof procedures by means of Labelled Deductive Systems and hopefully avoid the Caminada counter example.

We shall now show the idea behind this modification. Let us do the proof again, using our idea:

$$\begin{tikzcd}
A \arrow[d] & & \\
a \arrow[r] & b \arrow[r] & c \arrow[loop above]
\end{tikzcd}$$

Figure 21:

1. $a =$ in, assumption
2. $b =$ out, from (1)
3. we now have a node c which is not attacked by any node which is in, and instead of doing a case analysis, let us ask, by way of a subcomputation, can $c =$ in?

Subcomputation

- Given assumptions: $a =$ in, $b =$ out
- we test: $c =$ in.

(3.1) $c =$ in, assumption

(3.2) $c =$ out, from (3.1)

Therefore using the Shklop principle c surely must be out, and we move to Figure 22.

$$\begin{tikzcd}
 & & C \arrow[d] \\
a \arrow[r] & b \arrow[r] & c \arrow[loop above]
\end{tikzcd}$$

Figure 22:

(4) We now continue the original computation with Figure 22.

To make our idea crystal clear, let us present the reasoning structure as follows: (Note the network changes as we reason, so each line has to indicate which network we are dealing with).

1. Figure 20, $a =$ in, assumption
2. Figure 20, $b =$ out, from (1)
3. subcomputation in Box1

Box 1:

3.1	Figure 20, $c =$ in, assumption
3.2	Figure 22, $c =$ out, from (3.1)
3.3	Use (3.1) and (3.2) and the Shkop principle: c must be out
3.4	Exit subcomputation with Figure 22

4. We are now in Figure 22 from Box 1: we continue reasoning.

   Thus the network changes as we reason along the arrows!

At present we do not know if this new computation is sound. It may be that counter examples can be found. Even if it is sound, we do not know exactly what it does. Our conjecture is that it just forces us to consider the loops first and eliminate them. At any rate, this is not crucial to our paper, since we are happy with the General Shkop Principle and the algorithm we have in the Appendix.

4 Conclusion

The original Shkop principle is given in temporal context: Imagine a group of agents operating in time and taking actions. To execute an action **a** the pre-condition of the action $\alpha_{\mathbf{a}}$ needs to be fulfilled and then after the action is taken, the post-condition $\beta_{\mathbf{a}}$ of the action holds.

So if we start at time $t = 0$ in a certain state s and let our agents proceed with their actions then we move from state to state without any trouble and no argumentation networks arise and no loops arise.

The difference between ordinary actions and Talmudic actions is that the Talmud allows for the pre-conditions to contain future conditions and actions. Thus the enabling conditions of the actions can depend on the future and this can create loops. The Talmud also says that if the future condition is not fulfilled, then the action is nullified backwards in time (backward causality). See our papers [4] and [6].

To give a simple example, suppose that on Monday John orders a new laptop to be delivered on Friday. John gives his old laptop on Monday to a student named Tracy, free of charge, on the condition that on Friday, Tracy will configure his new laptop. Call this action **a**.

On Tuesday Tracy is ready to sell the laptop she got from John to a new buyer, Mary, for a good price, but Mary insists that on Friday Tracy transfers the contents of her old computer to the laptop she is buying. Call this action **b**.

The pre-condition for action **b** is that Tracy owns the laptop she is selling. For this to hold she must configure John's new laptop on Friday. However, if we allow action **b**, then Tracy will not be able to configure John's new laptop on Friday, because she will be busy transferring Mary's old data. If Tracy does that for Mary, then action **a** is nullified and so Tracy will not be the owner of the laptop she wants to sell and therefore action **b** is nullified.

What we get here is that if action **b** is allowed then it is nullified. The Shkop principle says that in this case do not allow action **b**.

We see here the context in which the Shkop principle operates. It is a time action model with future pre-conditions and backward causality, which progresses in time. Shkop says that any action which is about to be taken at time t which causes a chain reaction which cancels its own pre-condition at the same time t, should not be taken at time t.

We used the idea of Shkop to suggest and create the Shkop extensions for argumentation networks. These networks are not temporal but are static. We get them from the temporal action model by looking at what is happening at any certain fixed time.

This initial paper is mainly qualitative and a more detailed modelling of the temporal aspects is forthcoming.

Acknowledgements

We are grateful to Martin Caminada for helpful information and comments, and counter examples.

Appendices

A Tableaux for Caminada Socratic discussion

We offer here a tableaux method designed to test, for an element x in a finite argumentation network, whether x is an element of any admissible extension. Compare also with the Verheij paper [16].

Definition A.1. *Let $\mathbf{N} = (S, R)$ be a finite argumentation frame.*

1. *A tableaux for $\mathbf{N}$ has the form*
$$\tau = (\mathbb{A}_\tau, \mathbb{B}_\tau, \mathbb{D}_\tau)$$
where $\mathbb{A}_\tau \subseteq S$ is the left inside of τ and $\mathbb{B}_\tau \subseteq S$ is the right outside of τ, and $\mathbb{D}_\tau$ is the set of elements marked to be treated in τ. $\mathbb{D}_\tau$ will be treated in the next tableau derived from τ. We have either $\mathbb{D}_\tau \subseteq \mathbb{A}_\tau$ (left treatment) or $\mathbb{D}_\tau \subseteq \mathbb{B}_\tau$ (right treatment).

2. *A tableau τ is said to be closed if one or more of the following holds:*
   - $\mathbb{A}_\tau \cap \mathbb{B}_\tau \neq \varnothing$
   - *For some $y \in \mathbb{B}_\tau$, we have $\{x \in S | xRy\} = \varnothing$.*

Definition A.2. *Let $\mathbf{N} = (S, R)$ be finite argumentation frame and let $x \in S$. We define a tree $\mathbb{T}$ of tableaux for testing whether $x =$ in is possible at all, i.e. whether x can be a member of any admissible extension. The tree of tableaux will have tree relation ρ.*

Step 1
Form the tableau $\tau_1 \in \mathbb{T}$, where
$$\tau_1 = (\{x\}, \varnothing, \{x\})$$
say $\{x\}$ is marked to be dealt with at this stage.

Step 2
Form the tableau $\tau_2 \in \mathbb{T}$, where
$$\tau_1 = (\{x\}, \{y | yRx\}, \{y | yRx\})$$
Say $\{y | yRx\}$ are marked to be dealt with at this stage and that $\{x\}$ has been dealt with. Let $\tau_1 \rho \tau_2$ hold.

If for some y such that yRx we have $\{z | zRy\} = \varnothing$ or if xRx then this tableau is closed. Otherwise we move to Step 3.

Step 3
Let $\mathbf{f}$ be any choice function such that for each y to be dealt with in the tableaux τ_2 of the previous step, (i.e. $y \in \mathbb{D}_{\tau_2}$), it chooses an element $\mathbf{f}(y) \in S$ such that $\mathbf{f}(y)Ry$. Form the tableaux, $\tau_3^{\mathbf{f}} \in \mathbb{T}$:
$$\tau_3^{\mathbf{f}} = (\mathbb{A}_3^{\mathbf{f}}, \mathbb{B}_3^{\mathbf{f}}, \mathbb{D}_3^{\mathbf{f}})$$

for each such an $\mathbf{f}$*, where*

$$\begin{aligned}&\mathbb{A}_3^{\mathbf{f}} = \mathbb{A}_2 \cup \{\mathbf{f}(y) | y \in \mathbb{B}_2\}\\ &\mathbb{B}_3^{\mathbf{f}} = \mathbb{B}_2\\ &\mathbb{D}_3^{\mathbf{f}} = \{\mathbf{f}(y) | y \in \mathbb{B}_2 \textit{ and } \mathbf{f}(y) \notin \mathbb{A}_2\}.\end{aligned}$$

Say that all elements of $\mathbb{B}_2$ *(all the ys) have been dealt with and all elements of* $\mathbb{D}_3^{\mathbf{f}}$ *are marked to be dealt with.*

Let $\tau_2 \rho \tau_3^{\mathbf{f}}$*, for all* $\mathbf{f}$*.*

Note that $\mathbb{D}_3^{\mathbf{f}}$ *may be empty.*

Step 4
Let $\tau_3^{\mathbf{f}}$ *be any tableau of Step 3. Construct the tableau* $\tau_4^{\mathbf{f}} \in \mathbb{T}$ *as follows:*

$$\begin{aligned}&\mathbb{A}_4^{\mathbf{f}} = \mathbb{A}_3^{\mathbf{f}}\\ &\mathbb{B}_4^{\mathbf{f}} = \mathbb{B}_3^{\mathbf{f}} \cup \{z | \textit{ for some } u \in \mathbb{D}_3^{\mathbf{f}} \textit{ we have } zRu\}\\ &\mathbb{D}_4^{\mathbf{f}} = \{z | \textit{ for some } u \in \mathbb{D}_3^{\mathbf{f}} \textit{ we have } zRu \textit{ and } z \notin \mathbb{B}_3^{\mathbf{f}}\}.\end{aligned}$$

We say the elements of $\mathbb{A}_3^{\mathbf{f}}$ *have been dealt with and the elements of* $\mathbb{B}_4^{\mathbf{f}}$ *are marked to be dealt with.*

Let $\tau_3^{\mathbf{f}} R \tau_4^{\mathbf{f}}$*.*

Inductive step type odd
We assume by induction that we have $\tau = (\mathbb{A}, \mathbb{B}, \mathbb{D})$ *and the elements marked to be dealt with are all in* $\mathbb{A}$*, i.e.* $\mathbb{D} \subseteq \mathbb{A}$ *and* $\mathbb{D} \neq \varnothing$*. In this case proceed as in Step 3 and create* τ' *and let* $\tau' \in \mathbb{T}$ *and let* $\tau R \tau'$*.*

Inductive step type even
We assume by induction that we have $\tau = (\mathbb{A}, \mathbb{B}, \mathbb{D})$ *and all the elements to be dealt with are from* $\mathbb{B}$ *(i.e.* $\mathbb{D} \subseteq \mathbb{B}$*), and that* $\mathbb{D} \neq \varnothing$*.*

Then proceed as in Step 4.

Lemma A.3. *If* $\mathbf{N} = (S, R)$ *is finite then after a finite number of steps the Tableaux process terminates. We reach tableaux at the bottom of the* ρ*-tree such that they are either closed or their* $\mathbb{D}$ *is empty.*

Proof. Since $\mathbb{D}$ always adds new elments either to $\mathbb{A}$ or to $\mathbb{B}$ and $\mathbb{A}$ and $\mathbb{B}$ do not decrease, and S is finite, sooner or later $\mathbb{D} = \varnothing$. □

Lemma A.4. *Let* (S, R) *be a finite argumentation network and let* $(\mathbb{T}, \rho)$ *be the tableaux for it.*

Then there exists a maximal path $\tau_1 \rho \tau_2 \rho \ldots \rho \tau_n$ *of non-closed tableaux in* $\mathbb{T}$*, if and only if* x *is a member of some admissible extension* E*.*

Proof.

1. Assume $x \in E$ and E is an admissible extension. We will define a maximal path $\tau_1 \rho \tau_2 \rho \ldots \rho \tau_n$ of non-closed tableaux in $(\mathbb{T}, \rho)$.

   Let $\tau_1 = (\{x\}, \varnothing, \{x\})$ as in Step 1 of the inductive definition of $(\mathbb{T}, \rho)$.

   Let τ_2 be as in Step 2. τ_2 is not closed, because if xRx holds, then x cannot be in any admissible extension, and if for some y, yRx and $\neg \exists z(zRy)$ hold, then x is out.

   Assume by induction that we have defined a chain $\tau_1 \rho \tau_2 \rho \ldots \rho \tau_k$ of non-closed tableaux such that for each $1 \leq i \leq k$ we have

- If $y \in \mathbb{A}_{\tau_i}$ then $y \in E$
- If $y \in \mathbb{B}_{\tau_i}$ then for some $z \in E, zRy$ holds.

We now define τ_{k+1}.

Case k is odd
In this case we have

$$\mathbb{D}_{\tau_k} \subseteq \mathbb{A}_{\tau_k}$$

Let τ_{k+1} be defined in Inductive Step type odd (same as Step 3). Clearly $\tau_k \rho \tau_{k+1}$ holds. We want to show that τ_{k+1} is not closed. Since $\mathbb{A}_{\tau_k} \subseteq E$ and $\mathbb{D}_{\tau_k} \subseteq \mathbb{A}_{\tau_k}$ we have that any yRu for $u \in \mathbb{D}_{\tau_k}$ is atatcked by E and hence is out. Thus

$$\mathbb{A}_{\tau_{k+1}} \cap \mathbb{B}_{\tau_{k+1}} = \varnothing.$$

Also every such y is attacked by something and so τ_{k+1} is not closed.

Case k is even
In this case we have $\mathbb{D}_{\tau_k} \subseteq \mathbb{B}_{\tau_k}$. This means that all points of $\mathbb{D}_{\tau_k}$ are out. Moreover by construction, $\mathbb{D}_{\tau_k}$ are points attacking points in $\mathbb{A}_{\tau_{k-1}}$, and so by the admissibility of E each such point y has an attacker $\mathbf{f}(y) \in E$. Then let τ_{k+1} be $\tau^{\mathbf{f}}_{k+1}$ for this function $\mathbf{f}$. we have that $\tau_k \rho \tau^{\mathbf{f}}_{k+1}$ and $\tau^{\mathbf{f}}_{k+1}$ is non-closed.

We carry on until such an n that $\mathbb{D}_{\tau_n} = \varnothing$.

2. Assume there exists a maximal path of non-closed tableaux $\tau_1 \rho \tau_2 \rho \ldots \rho \tau_n$ in $(\mathbb{T}, \rho)$. Then clearly

$$\mathbb{D}_{\tau_n} = \varnothing.$$

Let $E = \mathbb{A}_{\tau_n}$. We show that E is conflict free and self-defending. If xRy holds for $x, y \in E$, then at some $\tau_i, y \in \mathbb{A}_{\tau_i}$ and so $x \in \mathbb{B}_{\tau_{i+1}}$ and so τ_j will be closed, for some $j \geq i$ (the j in which x gets into $\mathbb{A}_{\tau_j}$).

Assume for some z that $zRx, x \in E$. We need to show a $u \in E$ such that uRz. Since $x \in E$ then $x \in \mathbb{A}_{\tau_i}$ for some i. Then $z \in \mathbb{B}_{\tau_{i+1}}$ and so in $\mathbb{B}_{\tau_{i+1}} = \mathbb{B}_{\tau_i}\mathbf{f}$ we have $\mathbf{f}(z) \in \mathbb{A}_{\tau_i} = \mathbb{A}_{\tau_{i+1}}$ and $\mathbf{f}(z)Rz$.

This completes the proof. □

Example A.5. *Let us check again whether $c = in$ is possible in Figure 6, this time using tableaux.*

$$\begin{aligned} &\tau_1 : (\{c\}, \varnothing, \{c\}) \\ &\tau_2 : (\{c\}, \{a, b\}, \{a, b\}) \\ &\tau^{\mathbf{f}}_3 : (\{c, a, b\}, \{a, b\}, \{a, b\}). \end{aligned}$$

Here $\mathbf{f}(a) = b$ and $\mathbf{f}(b) = a$. $\tau^{\mathbf{f}}_3$ is closed.

Remark A.6. *Note that the tableaux method works for the query for several points, namely*

- *Can $c_1, \ldots, c_n$ all be together in some admissible set?*

We simply start our tableaux with

Step 1:

$$(\{c_1, \ldots, c_n\}, \varnothing, \{c_1, \ldots, c_n\})$$

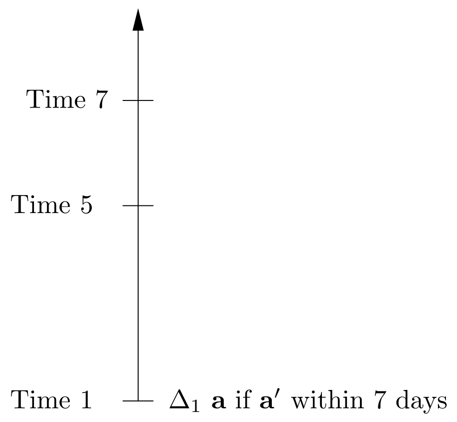

Figure 23:

B Shkop principle in temporal context

It would be helpful to the reader if we present the Shkop prinicple in its natural temporal context. Imagine a linear flow of time of the form $(N, <)$ where N is the set of natural numbers $\{1, 2, 3, \ldots\}$ and $<$ is smaller than relation. We associate with each $n \in N$, a state of the world, which we denote by Δ_n, being a classical propositional logical theory in the language with the atoms $Q = \{q_1, \ldots, q_k\}$. We imagine history as evolving. At step 1 we have only state Δ_1 as given and state Δ_2 has not been created yet. The future states are created by actions. An action has the form $\mathbf{a} = (\alpha_{\mathbf{a}}, \beta_{\mathbf{a}})$, where $\alpha_{\mathbf{a}}$ is the precondition of the action and $\beta_{\mathbf{a}}$ is the post condition, all in the same classical langauge of the states Δ.

So at state Δ_1 we might wish to take action **a**. We can do that if the precondition holds, i.e. $\Delta_1 \vdash \alpha_{\mathbf{a}}$. If this is the case, then we take the action and we move to state Δ_2, which is the state at time 2. Δ_2 is connected with Δ_1 via a revision process, denoted by "$\circ$". Thus $\Delta_1 = \Delta_1 \circ \beta_{\mathbf{a}}$.

The exact nature of the revision process is not relevant to our purpose. It is sufficient to know that for any Δ and any action $\mathbf{a} = (\alpha_{\mathbf{a}}, \beta_{\mathbf{a}})$, such that $\Delta \vdash \alpha_{\mathbf{a}}$ we get a new state $\Delta' = \Delta \circ \beta_{\mathbf{a}}$.

This is a simple model which can easily be made richer and more complicated. The Talmudic twist to this model is that the Talmud allows for future *conditional actions*. Part of the precondition for allowing action **a** to take place at time n is that a related action $\mathbf{a}'$ be taken at future time $n + n'$.

For example: I give you this computer to be yours now on the condition that you clean my garden in a week's time. We have

$\mathbf{a} =$ (I own computer, you own computer)
$\mathbf{a}' =$ (Truth, you have cleaned my garden)

So if you do not clean my garden in a week, then the original action is cancelled. This is *backward causality.* We denote these conditional actions by **a** if a' within n' days.

Consider now Figure 23.

So action **a** can be taken at Δ_1 on the condition that action $\mathbf{a}'$ is taken at time 7. Note that time 7 has not yet happened.

Now suppose we continue to take actions and at time 5 we want to take action **b**. The precondition of action **b** holds and so we want to proceed. It could be the case that if we take action **b** at time 5, then a situation is created where action $\mathbf{a}'$ cannot be taken at time 7. If action $\mathbf{a}'$ is not taken at time 7, then action **a** at time 1 is not valid and past history is affected to the extent that at time 5 in the new history, the precondition for action **b** does not hold. The Shkop prinicple says that any action **b**, which when taken, changes history backwards in such a way that it cannot be taken (cancelling its own precondition) then **b** should not be taken!

A simple example will illustrate the idea:

Example B.1. *On Monday, John buys a new computer to replace his old one. He gives the old computer (which is still good and fast) to his student Terry, on the condition that on Saturday, Terry comes to John's home and installs the new computer.*

Terry decides to sell the computer he was given to a housewife neighbour called Mary. The precondition for the sale is that Terry owns the computer. This is OK because there is still the possibility for Terry to fulfil the condition to John and go on Saturday and install John's new computer.

Mary is prepared to buy the computer from Terry but she has her own condition. She wants Terry to come on Saturday and teach her how to use it.

We ask: can Terry sell the computer to Mary? We reason, following Shkop, that if Terry does sell the computer to Mary, he will have to spend Saturday with her and would not be able to go to John and install John's new computer. Failing to go to John on Saturday would nullify the gift of John giving Terry the old computer, which would nullify the precondition of the sale of this computer by Terry to Mary, namely Terry is not the owner of this computer.

So by selling the computer to Mary, Terry is nullifying the legitimacy of the sale!

So the Shkop prinicple applies and the sale cannot be permitted.

Let us now give another temporal argumentation model in which the Shkop principle can apply for resolving loops.

Suppose we have a sequence of argumentation networks of the form $(S_n, R_n), n = 1, 2, 3, \ldots$ such that $S_n \subseteq S_{n+1}$ and $R_n \subseteq R_{n+1}$. Thus as time passes on, (i.e. $n = 1, 2, \ldots$) we get more and more arguments and more and more attacks.

Consider time $n+1$ and let $x \in S_{n+1} - S_n$. So x is a new argument added at time $n+1$. So if x causes an odd loop and cannot be part of any extension, then we apply the Shkop principle, as detailed in Section 2, and annihilate it. To understand the usefulness of this principle and the temporal setup, consider Figure 13. We have many options for resolving the loops there. Our task is made easier if we have a temporal sequence of when each argument was put into the figure. We can follow the temporal sequence and use the Shkop principle to incrementally in time resolve the loops.

References

[1] M. Abraham, D. Gabbay, and U. Schild. Analysis of the Talmudic Argumentum A Fortiori Inference Rule (Kal-Vachomer) using Matrix Abduction. *Studia Logica*, vol 92 , No 3, pp 281–364, 2009.

[2] M. Abraham, G. Hazut, D. Gabbay, Maruvka and U. Schild. Logical Analysis of the Talmudic Rule of General and Specific (Klal-u-Prat). Special issue on Judaic Logic edited by A Schumann, *Journal of the History and Philosophy of Logic*, Vol 32 issue 1, pp 47–62, 2011.

[3] M. Abraham, D. Gabbay, and U. Schild. Obligations and Prohibitions in Talmudic Deontic Logic. In G. Governatori and G. Sartor, eds., *DEON 2010, LNAI 6181*, pp. 166–178, 2010.

[4] M. Abraham, D. Gabbay, and U. Schild. Contrary to Time Conditionals in Talmudian Law. *Journal of Artificial Intelligence and Law*, 20:2, 145–179, 2012.
Submitted to *J of AI and Law.*

[5] M. Abraham, D. Gabbay, and U. Schild. Obligations and Prohibitions in Talmudic Deontic Logic. *Journal of Artificial Intelligence and Law*, 19:2, 117–148, 2011.

[6] M. Abraham, I. Belfer, D. Gabbay, and U. Schild. Future determination of entities in Talmudic Logic. Submitted.

[7] D. Gabbay. Equational Approach to Argumentation Networks. *Argument and Computation*, 3:2-3, 87-142, 2012.

[8] S. A. Gaggl and S. Woltran. Strong Equivalence for Argumentation Semantics Based on Conflict-Free Sets, slides of lecture presented in ECSQARU 2011.

[9] H. Prakken. An abstract framework for argumentation with structured arguments *Argument & Computation*, 2010.

[10] M. Caminada. Preferred Semantics as Socratic Discussion, `http://homepages.abdn.ac.uk/martin.caminada/pages/publications/preferred_game.pdf`, 2010.

[11] T. Wakaki. Preference-based Argumentation Capturing Prioritized Logic Programming, *ArgMAS*, 2010.

[12] M. Caminada, Y. Wu. On the Limitations of Abstract Argumentation. `http://users.numericable.lu/martincaminada/publications/BNAIC_limitations_abstract.pdf`

[13] P. Baroni and M. Giacomin. Solving semantic problems with odd-length cycles in argumentation. In *Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2003)*, pp. 440–451. LNAI 2711, Springer-Verlag, Aalborg, Denmark, 2003.

[14] G. A. Bodanza and F. A. Tohmé. Two approaches to the problems of self-attacking arguments and general odd-length cycles of attack. *Journal of Applied Logic*, 7, 403–420, 2009.

[15] P. Baroni, M. Giacomi and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168 (1-2):162–210, 2005.

[16] Bart Verheij. A Labelling approach to to the computation of credulous acceptance in argumentation. In *Proceedings IJCAI'07. Proceedings of the 20th international joint conference on Artifical intelligence*, 623–628. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2007.

[17] Martin Caminada. A Labelling Approach for Ideal and Stage Semantics. *Argument and Computation*, 2 (1): 1–21, 2011.