UNIVERSITÄT KARLSRUHE

FAKULTÄT FÜR INFORMATIK

Postfach 6980, D-76128 Karlsruhe,
Deutschland

A Verification Concept for SDL Systems and its Application to the Abracadabra Protocol

Markus Rinderspacher

Institut für Logik, Komplexität und Deduktionssysteme

Interner Bericht 14/94

Abstract

SDL is a specification language to specify distributed systems. Especially it is suitable for communication protocols. In some cases however it is not enough to describe just the behaviour of a protocol, but there are formulated some additional properties as requirements of the SDL system. A formalism convenient to describe them is for example first order logic. Our approach is to prove such properties with methods of automated reasoning after transforming the SDL specification into a first order logic specification. The proofs are done with the program verification system *Tatzelwurm*, especially with its prover.

Practical experience shows that it is convenient to do a proof in two steps. In the first step the behaviour of the system is calculated out of the behaviour of the agents. The proofs of this step is independent of the property to prove. In this report we give a proof methods containing instructions how the arguments are applied during these proofs. It is shown how reachability analysis is done during a formal proof and how fairness arguments are applied.

The report contains two papers, where the first one describes the formal basis of the method and shows the proof obligations occurring verifying a communication protocol. The second paper shows how some tedious tasks can be done more elegant using rewrite rules and recursive equations.

In the appendix we give two examples out of the verification of the Abracadabra Protocol.

Contents

Appendix:

1. Proof Structuring with Fixed Point Theory

Abstract

Starting from a purely functional description of a communication protocol, we present a method how correctness proofs including safety- and progress properties can be developed systematically with an automatic theorem prover. We show how a complex proof can be divided into smaller ones due to proof arguments typically occurring in the area of protocol verification. Experience with this method shows that proofs can be developed with an acceptable amount of work.

1.1. Introduction

The objective we pursue with our work is to verify properties of communication protocols. We start with an automata based description of the protocol and a property *p* given in terms of first-order logic. Due to Broy [Bro87,91] we model the behaviour of the protocol agents with functions over streams and the system behaviour with its least fixed point *c*. If we use this formalism, we can reduce the verification problem to a proof of the theorem

$$\mathrm{lf}(c) \rightarrow \mathrm{p}(c).$$

lf is a logical formula formalizing that *c* is the least fixed point of the system regarded. To find the proof of this theorem we use the program verification system *Tatzelwurm* [KÄU89$_1$] with its integrated tableau prover. This prover is designed to prove formulae occurring in program verification. It is enlarged by decision procedures [KÄU89$_2$] for theories typically occurring in the verification of programs and protocols, like arithmetic, records and lists. Furthermore *Tatzelwurm* supports interactive theorem proving with a powerful user-interface. Interactive proving is necessary, since automatic proofs of such complex problems seem to be impossible because of the enormous search space, which must be managed.

Nevertheless interactive theorem proving is not the solution of all problems. If we try to find a proof for the above theorem, we will get completely lost, if we have no method how to find this proof.

The presented method is part of our experience with the verification of the *Abracadabra* protocol [Isc89]. There we took a paper [Bro87], which suggested, how the behaviour of the Abracadabra-protocol can be modelled and how its correctness proof can be found. We did this proof with *Tatzelwurm* and were successful after we had made some essential changes to the suggestions of Broy.

Respect to our experience we suggest for future verification projects two steps:

(1) calculate the fixed point of the system explicitly and

(2) prove the property.

Calculation of the fixed point is a subgoal occurring in every protocol verification project. It causes a lot of work, so that here a method reduces proof expense.

We organize this paper by giving in Section 2 some results of fixed point theory and especially

the fixed point theorem, the basis of our reasoning.
In Section 3 we will give a formalization of a protocol and show that a correctness proof can be systematically developed applying well-known arguments like reachability and fairness.
Section 4 collects some experience obtained proving with *Tatzelwurm* and finally Section 5 concludes the results and gives hints, how to continue in future.

1.2. Fixed points of Functions over Lists

In this section we want to give some results of fixed point theory, which is the basis for our later argumentation. We regard fixed points in complete partial orders. The following definitions and theorems can be found in [Loe87].
Let $(S, \leq)$ be a partial order, then an element $m \in S$ is called the *least (greatest)* element of a set S, if $m \leq s$ $(s \leq m)$ for all $s \in S$

Definition 1.2.1.: (Least Upper Bounds)
Let $(D, \leq)$ be a partial order and S a (possibly empty) subset of D. An element $u \in D$ is said to be an *upper bound* of S (in D), if $d \leq u$ for all $d \in S$; u is said to be the *least upper bound* (lub) of S (in D), if u is the least element of the set of all upper bounds of S in D.

Note that the least upper bound is uniquely determined, if it exists. In the following a totally ordered subset S of D is called a *chain*.

Definition 1.2.2.: (Complete Partial Orders)
A partial order $(D, \leq)$ is a *complete partial order (cpo)*, if the following two conditions hold:
1. The set D has a least element.
2. For every chain S in D the least upper bound $\sqcup S$ exists.

Definition 1.2.3.: (Continuity)
Let $(D, \leq)$ and $(E, \leq)$ be cpo's. A function $f: D \rightarrow E$ is said to be *continuous*, if for every chain S in D, $f(\sqcup S) = \sqcup f(S)$.

Theorem 1.2.4.: (The Fixed Point Theorem)
Let $(D, \leq)$ be a cpo and $f: D \rightarrow D$ be a continuous function. Then f has a (uniquely determined) least fixed point $mf = \sqcup\{f^i(\perp) \mid i \in Nat\}$, where $\perp$ is the least element of the cpo $(D, \leq)$.

Now we want to apply the fixed point theorem to lists and functions over lists. The following theorems are easy to prove. Let G be a set of objects and let L(G) be the possibly infinite lists with elements of G. The application of the fixed point theorem demands cpos and continuous functions. We have to add a least element ω to G and obtain a flat partial order $(G^{\omega}, \leq)$. We define a prefix ordering « on the lists and add an element ε, which is called the *empty* list. We consider these lists $L(G^\omega)$ as a subset of the function set $((Nat \rightarrow G^\omega), \leq)$, which is known to be a cpo. The following theorem establishes that $(L(G^\omega), «)$ is a cpo. We simplify our notation in writing L instead of $L(G^\omega)$, assuming the element sort of the list is a flat partial order.

Theorem 1.2.5.: $(L(G^\omega), «)$ is a sub-cpo of $((Nat \rightarrow G^\omega), \leq)$

Now we study the continuity of functions in more detail. We need some criteria, which allow to check, whether the functions are continuous.
At first we examine the standard list functions cons, car, cdr and an additional function app, which appends a list to a finite list.

Theorem 1.2.6.: The functions cons, car, cdr are continuous in their list argument. The function app is continuous in its second argument.

We write in the following *&* instead of *cons* and *app*. Examining the arguments allows us to determine, which of the constructors is meant. As shortcut we write m^k for the list having exactly *k* elements *m*. Later we will describe processes by functions over lists. The following theorem allows us to construct continuous functions using continuous functions and some conditions. It will be used to establish the continuity of the functions defined 3.3 and 3.4.

Theorem 1.2.7.:
Let I be a finite index set and $i, j \in I$. Let $\{f_i \mid i \in I\} \subset [L \rightarrow L]$ be a finite set of continuous functions and $\{h_i \mid i \in I\} \subset [G \rightarrow G]$. Let $B_i \subset L$, where $B_i \cap B_j = \emptyset$ and $\bigcup B_i = L$, with:
if $l « l'$ and $l \in B_i$ then $l' \in B_i$.
By the following a continuous function f is defined:
(1) $f(\varepsilon) = \varepsilon$
(2) $f(cons(g, l)) = cons(h_i(g), f_i(l))$ if $cons(g, l) \in B_i$ for all $i \in I$

1.3. Development of a Correctness Proof

In this section we want to show, how fixed point theory can be used to develop correctness proofs systematically. The example used here is taken out of the verification of the Abracadabra-protocol [Isc21], which operates over a full-duplex communication medium between two agents. The medium may occasionally lose messages, but will not disorder, corrupt, invent or duplicate messages.
The two stations communicate transferring Protocol Data Units (PDUs). The service, provided by the protocol must be reliable, in the sense that the protocol guarantees the transmission of the data, coming from the user of the service. We assume, that we know in advance which of agents is the *sender* and the *receiver* respectively. Also we split up the bidirectional medium into

two unidirectional media *mu*, having the same behaviour.
Fig 1.3.1. shows the system regarded in this paper, where the names in brackets are the functions describing the behaviour of the agents. The protocol has to transmit a sequence *s* of data. We have to proof that this sequence is eventually transmitted.

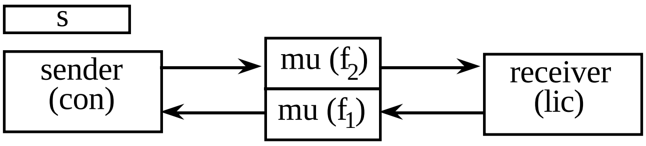

Fig. 3.1. The system regarded

The first problem, to be solved is the calculation of the system behaviour using the behaviour of the cooperating agents. To transmit the data the agents send and receive *messages*. Lists of messages are called *streams*. We describe the behaviour of the agents with continuous functions over streams, called *stream processing functions*. Then we define the behaviour of the system by a recursion equation, which has a solution due to the fixed point theorem.
The sender transmits *s* to the receiver by sending a stream *c*(*s*), which depends beside s on the implementation of the sender and on the input it gets from the medium. Due to [Bro87] *c*(*s*) is the solution of the equation $c(s) = con(f_1(lic(f_2(c(s)))))$. Due to the fixed point theorem the solution is uniquely determined, provided the functions are continuous.

To define the behaviour of the agents we assume that a message is one of the in pairs different objects {*em*, *cr*, *cc*, *dt*, *ak*, *dr*, *dc*}. It is *dt* a two place, *ak* a one place function symbol and the rest are constant symbols. *dt* takes as argument a data and a control-bit and *ak* a control-bit. *em* is used to model, that the medium does not deliver one of the other messages. The control-bit is assumed to be one of the boolean objects {*tt*, *ff*}. We have a function symbol *non*, to define a function mapping *tt* to *ff* and vice versa. In the following are *m*, *m'* arbitrary messages, *c*, *c'* streams, *d*, *d'* data, *s*, *s'* list of data and *a*, *b* boolean.

Definition 1.3.1.: (Medium Function)
A medium function f: L → L is a function having the following properties:

1 f(ε) = ε

2. car(f(m & c)) = m or car(f(m & c)) = em
3. f(m & c) = f(m & ε) & f(c)

Lemma 1.3.2: A medium function is continuous

Proof: (a sketch) It is a well-known fact, that it is enough to prove that f is monotone and
f(⊔S) « ⊔f(S) for every chain S with an infinite number of elements

1. Monotony: Let c « c'. If c is infinite, c' is also infinite and c = c'. Then f(c) = f(c'), which is sufficient for f(c) « f(c').
   If c is a finite stream, we can write c' = c & c''. Since f(c) « f(c) & f(c''), f(c) « f(c') holds

2. $f(\sqcup S)$ « $\sqcup f(S)$: Let S = $\{c_i \mid i \in$ Nat$\}$ be a chain. Then f(S) := $\{f(c_i) \mid i \in$ Nat$\}$ is also a chain. Since S and f(S) have an infinite number of elements, so $\sqcup$S and $\sqcup$f(S) are infinite streams. Since f($\sqcup$S) is an infinite stream and $\sqcup$f(S) « f($\sqcup$S), $\sqcup$f(S) = f($\sqcup$S) holds.

We show a sketch of the behaviour of the system in Fig. 1.3.1. It is shown by the following state transition diagrams and the parallel operator ‖. The state labeled with x means termination and the transitions are labeled with pairs (i; o) of I/O-signals, where - means no input (output). We denote the indeterminism of the medium by τ-transitions. The other indeterminisms are occurring, because the diagram does not describe all the details. It is omitted that the agents‘ *sender* and *receiver* have an additional boolean control variable, which makes them deterministic.

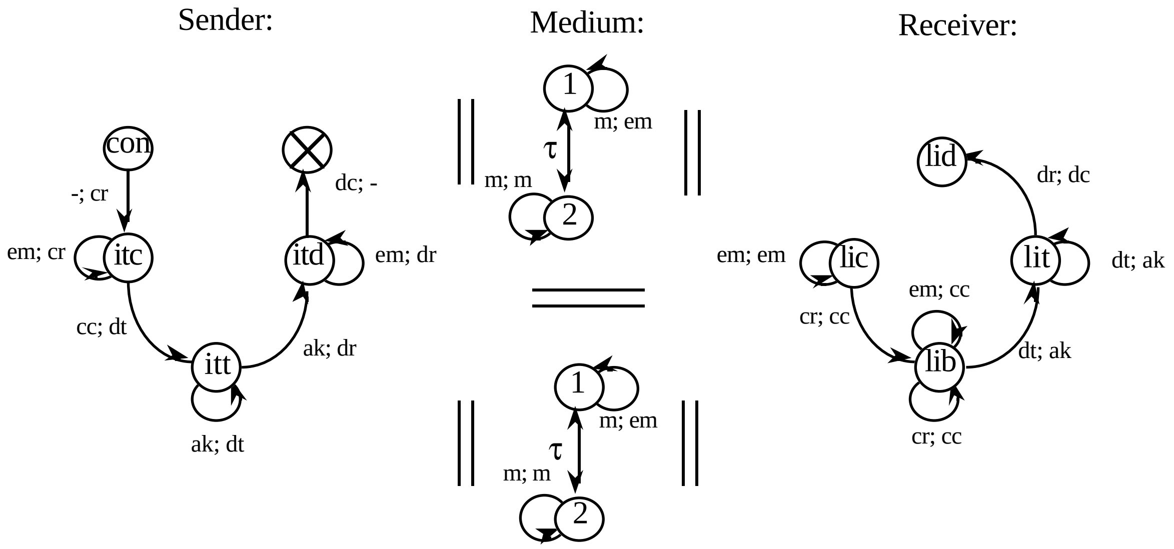

Fig. 1.3.2. The components of the system

The full behaviour of these agents is given by the following functions. For each state we define a function typically having three arguments, the input stream, the data to be transmitted, and the state of the control variable. If arguments are not necessary, we omit them.

Definition 1.3.3: (Sender)

A sender-function *con* is a function, having the following properties:

(Con) con(c, s) = cr & itc(c, s)

(Itc_1) itc(cc & c, d & s) = dt(d, tt) & itt(c, d & s, tt)

(Itc_2) itc(em & c, s) = cr & itc(c, s)

(Itt_1) itt(ak(a) & c, d & d’ & s, a) = dt(d’, non(a)) & itt(c, d’ & s, non(a))

(Itt_2) itt(ak(a) & c, d & ε, a) = dr & itd(c)

(Itt_3) (em = m $\vee$ cc = m $\vee$ m = ak(non(a)))

$\rightarrow$ itt(m & c, d & s, a) = dt(d, a) & itt(c, d & s, a)

(Itd_1) $\mathrm{itd}(dc \;\&\; c) = \varepsilon$

(Itd_2) $\neg\, dc = m \rightarrow \mathrm{itd}(m \;\&\; c) = dr \;\&\; \mathrm{itd}(c)$

(Id) For the rest of the streams the functions are the identity on their stream.

Definition 1.3.4: (Receiver)
A receiver-function *lic* is a function, having the following properties:

(Lic_1) $\mathrm{lic}(cr \;\&\; c) = cc \;\&\; \mathrm{lib}(c)$

(Lic_2) $\neg\, cr = m \rightarrow \mathrm{lic}(m \;\&\; c) = em \;\&\; \mathrm{lic}(c)$

(Lib_1) $\mathrm{lib}(dt(d, a) \;\&\; c) = \mathrm{lit}(dt(d, a) \;\&\; c, tt)$

(Lib_2) $(cr = m \vee em = m) \rightarrow \mathrm{lib}(m \;\&\; c) = cc \;\&\; \mathrm{lib}(c)$

(Lit_1) $\mathrm{lit}(dt(d, a) \;\&\; c, a) = ak(a) \;\&\; \mathrm{lit}(c, non(a))$

(Lit_2) $\mathrm{lit}(dr \;\&\; c, a) = \mathrm{lid}(dr \;\&\; c)$

(Lit_3) $m = dt(d, non(a)) \vee em = m \rightarrow \mathrm{lit}(m \;\&\; c, a) = ak(non(a)) \;\&\; \mathrm{lit}(c, a)$

(Lid_1) $\mathrm{lid}(dr \;\&\; c) = dc \;\&\; \mathrm{lid}(c)$

(Lid_2) $\neg\, dr = m \rightarrow \mathrm{lid}(m \;\&\; c) = dc \;\&\; \mathrm{lid}(c)$

(Id) For the rest of the streams the functions are the identity on their stream.

Note that these functions are continuous respect to 1.2.7. We have described the system behaviour with stream-processing functions of its components and the equation $c(s) = con(f_1(lic(f_2(c(s)))))$. To prove properties however we have to solve this equation, which is the same as to calculate the fixed point explicitly.

We do this by manipulating the fixed point equation in a systematic way. In our example we define 6 more equations and fixed points and 16 theorems describing the relations between these fixed points.

At first we give these equations and theorems and then we show using an example how we get the fixed point equations and the theorems successively.

We define fixed points by following equations:

(F_0) $c(s) = con(f_1(lic(f_2(c(s)))))$

(F_1) $r_1(s) = itc(f_1(cc \;\&\; lib(f_2(r_1(s)))), s)$

(F_2) $s \neq \varepsilon \rightarrow r_2(s) = itt(f_1(lib(f_2(dt(car(s), tt) \;\&\; r_2(s)))), s, tt)$

(F_3) $s \neq \varepsilon \rightarrow r_3(s, a, non(a)) = itt(f_1(ak(non(a)) \;\&\; lit(f_2(r_3(s, a, non(a))), a)), s, non(a))$

(F_4) $s \neq \varepsilon \rightarrow r_4(s, a, a) = itt(f_1(lit(f_2(dt(car(s), a) \;\&\; r_4(s, a, a)), a)), s, a)$

(F_5) $r_5(a) = itd(f_1(lit(f_2(dr \;\&\; r_5(a)), a)))$

(F_6) $r_6 = itd(f_1(dc \;\&\; lid(f_2(r_6))))$

The relations between the fixed points are given by the following theorems:

(S_1) $c(s) = cr \;\&\; r_0(s)$

(S_2) $car(f_2(cr \;\&\; r_0(s))) \neq em \rightarrow r_0(s) = r_1(s)$

(S_3) $car(f_2(cr \;\&\; r_0(s))) = em \rightarrow r_0(s) = cr \;\&\; r_0(s)$

(S_4) $s \neq \varepsilon \wedge car(f_1(cc \;\&\; lib(f_2(r_1(s))))) \neq em \rightarrow r_1(s) = dt(car(s), tt) \;\&\; r_2(s)$

(S_5) $car(f_1(cc \,\&\, lib(f_2(r_1(s))))) = em \rightarrow r_1(s) = cr \,\&\, r_1(s)$

(S_6) $car(f_2(dt(car(s), tt) \,\&\, r_2(s))) \neq em \rightarrow r_2(s, i, k) = r_3(s, ff, tt)$

(S_7) $s \neq \varepsilon \wedge car(f_2(dt(car(s), tt) \,\&\, r_2(s))) = em \rightarrow r_2(s) = dt(car(s), tt) \,\&\, r_2(s)$

(S_8) $s \neq \varepsilon \wedge cdr(s) \neq \varepsilon \wedge car(f_1(ak(non(a)) \,\&\, lit(f_2(r_3(s, a, non(a))), a))) \neq em$
$\rightarrow r_3(s, a, non(a)) = dt(car(cdr(s)), a) \,\&\, r_4(cdr(s), a, a)$

(S_9) $s \neq \varepsilon \wedge car(f_1(ak(non(a)) \,\&\, lit(f_2(r_3(s, a, non(a))), a))) = em$
$\rightarrow r_3(s, a, non(a)) = dt(car(s), non(a)) \,\&\, r_3(s, a, non(a))$

(S_{10}) $s \neq \varepsilon \wedge car(f_2(dt(car(s), a) \,\&\, r_4(s, a, a))) \neq em \rightarrow r_4(s, a, a) = r_3(s, non(a), a)$

(S_{11}) $s \neq \varepsilon \wedge car(f_2(dt(car(s), a) \,\&\, r_4(s, a, a))) = em$
$\rightarrow r_4(s, a, a) = dt(car(s), a) \,\&\, r_4(s, a, a)$

(S_{12}) $s \neq \varepsilon \wedge cdr(s) = \varepsilon \wedge car(f_1(ak(non(a)) \,\&\, lit(f_2(r_3(s, a, non(a))), a))) \neq em$
$\rightarrow r_3(s, a, non(a)) = dr \,\&\, r_5(a)$

(S_{13}) $car(f_2(dr \,\&\, r_5(a))) \neq em \rightarrow r_5(a) = r_6$

(S_{14}) $car(f_2(dr \,\&\, r_5(a))) = em \rightarrow r_5(a) = dr \,\&\, r_5(a)$

(S_{15}) $car(f_1(dc \,\&\, lid(f_2(r_6)))) \neq em \rightarrow r_6 = \varepsilon$

(S_{16}) $car(f_1(dc \,\&\, lid(f_2(r_6)))) = em \rightarrow r_6 = dr \,\&\, r_6$

As an example we show how we can deduce the fixed point equation F_2 and the theorems S_4 and S_5 from F_1.

(F_1) $r_1(s) = itc(f_1(cc \,\&\, lib(f_2(r_1(s)))), s)$
Due to the indeterminism of the medium we have to treat two cases.

1. Suppose $f_1(cc \,\&\, lib(f_2(r_1(s)))) = cc \,\&\, f_1(lib(f_2(r_1(s))))$.
   If we assume, that $s \neq \varepsilon$, we get with the definition (Itc_1) of 1.3.3.
   $r_1(s) = dt(car(s), tt) \,\&\, itt(f_1(lib(f_2(r_1(s)))), s, tt)$, which leads to
   $cdr(r_1(s)) = itt(f_1(lib(f_2(dt(car(s), tt) \,\&\, cdr(r_1(s))))), s, tt)$,
   which is a fixed point equation, which has a least fixed point, we name it $r_2(s)$. We get:
   (F_2) $s \neq \varepsilon \rightarrow r_2(s) = itt(f_1(lib(f_2(dt(car(s), tt) \,\&\, r_2(s))), s, tt)$.
   Since the least fixed point is uniquely determined, it is $r_1(s) = dt(car(s), tt) \,\&\, r_2(s)$.
   We get (S_4) by collecting the assumptions we made.
2. We assume $f_1(cc \,\&\, lib(f_2(r_1(s)))) = em \,\&\, f_1(lib(f_2(r_1(s))))$.
   With the definition (Itc_2) of 1.3.3. we get
   $r_1(s) = cr \,\&\, itc(f_1(lib(f_2(r_1(s)))), s)$, which leads to
   $cdr(r_1(s)) = itc(f_1(lib(f_2(cr \,\&\, cdr(r_1(s))))), s)$
   Applying the definition of the medium and (Lib_2) of 1.3.4. we get
   $cdr(r_1(s)) = itc(cc \,\&\, f_1(lib(f_2(cdr(r_1(s))))), s)$
   Since the least fixed point is uniquely determined, it´s $r_1(s) = cr \,\&\, r_1(s)$.
   We get (S_5) by collecting the assumptions we made.

We received the theorems S_1, ..., S_{16} by manipulating of the fixed point equations. We illustrate them in the following diagram graphically. We got the insight that manipulation of fixed point equations in the described way is a reachability analysis during a proof. The fixed points can be regarded as states and the relations between the fixed points can be viewed as reachability graph. In this case the graph describes the behaviour on the senders point of view. The sender is not able to recognize, whether the medium described by f_2 transmits the signal or not. This is reflected by the τ-transition in the diagram.

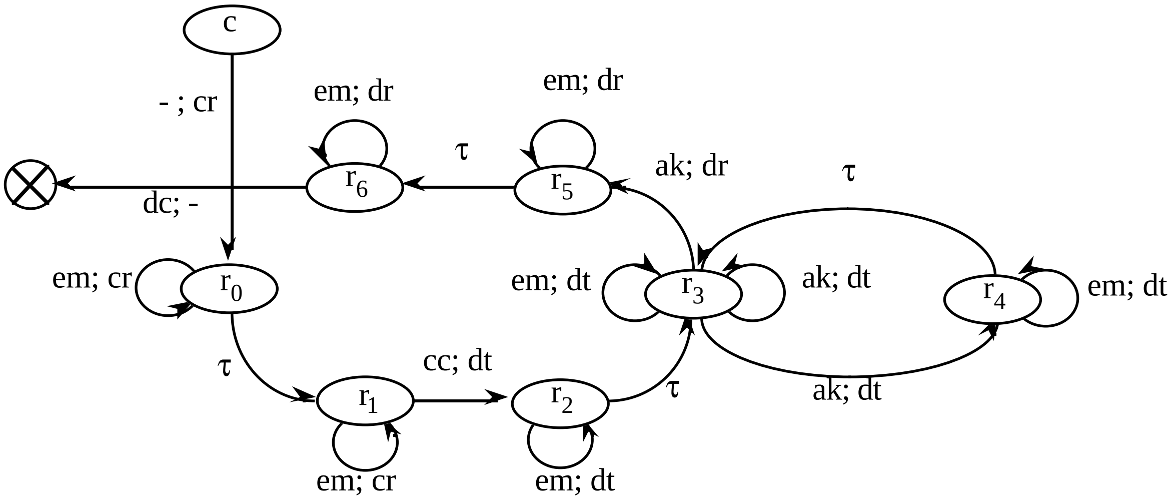

Fig. 1.3.3. Relations between the fixed points

We have examined the system behaviour without fairness until now. Fairness should guarantee that the system eventually leaves a state after entering it. We systematically derive a formulation of the system behaviour including fairness. To speak in terms of Fig. 1.3.3. we show that the small loops eventually terminate, provided that the medium is *fair*.

Definition 1.3.5: (Fair Medium Function)
A fair medium function f is a medium function with the following additional property
$\exists$ k. $f(m^{k+1} \& c) = f(em^k \& m \& c)$

With this property we are able to prove following theorems, which enlarges the first formalization of the system to a formalization containing liveness.

(S_a) $\exists$ k. $c(s) = cr^{k+1} \& r_1(s)$

(S_b) $s \neq \varepsilon \rightarrow \exists$ k. $r_1(s) = cr^k \& dt(car(s), tt) \& r_2(s)$

(S_c) $s \neq \varepsilon \rightarrow \exists$ k. $dt(car(s), tt) \& r_2(s) = dt(car(s), tt)^{k+1} \& r_3(s, ff, tt)$

(S_d) $s \neq \varepsilon \wedge cdr(s) \neq \varepsilon$

$\rightarrow \exists$ k. $r_3(s, a, non(a)) = dt(car(s), non(a))^k \& dt(car(cdr(s)), a) \& r_4(cdr(s), a, a)$

(S_e) $s \neq \varepsilon \rightarrow \exists$ k. $dt(car(s), a) \& r_4(s, a, a) = dt(car(s), a)^{k+1} \& r_3(s, non(a), a)$

(S_f) $s \neq \varepsilon \wedge \mathrm{cdr}(s) = \varepsilon \rightarrow \exists\, k.\ r_3(s, a, \mathrm{non}(a)) = \mathrm{dt}(\mathrm{car}(s), \mathrm{non}(a))^k\ \&\ \mathrm{dr}\ \&\ r_5(a)$
(S_g) $\exists\, k.\ \mathrm{dr}\ \&\ r_5(a) = \mathrm{dr}^{k+1}\ \&\ r_6$
(S_h) $\exists\, k.\ r_6 = \mathrm{dr}^k$

It is now possible to prove properties of the formalized system. We prove properties, which are formalized by requirements on fixed points with induction over the list s. Induction over lists demands that the lists are finite. Note that the loop consisting of state r_3 and r_4 is terminating, if the list s is finite.

We proof now by induction the property, that the sender transmits all the data of s. Let $\underline{k}$ be a list of integers. To express a property at first we define functions *messc* and *messt,* which extract the transmitted data out of a list of messages. *messc* and *messt* have the following properties:

(1) $\mathrm{messc}(\varepsilon) = \mathrm{messt}(\varepsilon, a) = \varepsilon$
(2) $\mathrm{messt}(\mathrm{dt}(d, a)\ \&\ c, a) = d\ \&\ \mathrm{messt}(d, \mathrm{non}(a))$
(3) $\mathrm{messt}(\mathrm{dt}(d, \mathrm{non}(a))\ \&\ c, a) = \mathrm{messt}(\mathrm{dr}\ \&\ c, a) = \mathrm{messt}(c, a)$
(4) $\mathrm{messc}(\mathrm{cr}\ \&\ \mathrm{cr}\ \&\ c) = \mathrm{messc}(\mathrm{cr}\ \&\ c)$
(5) $\mathrm{messc}(\mathrm{cr}\ \&\ \mathrm{dt}(d, a)\ \&\ c) = \mathrm{messt}(\mathrm{dt}(d, a)\ \&\ c, \mathrm{tt})$
(6) $\mathrm{messc}(\mathrm{cr}\ \&\ \mathrm{dr}\ \&\ c) = \mathrm{messt}(\mathrm{dr}\ \&\ c, \mathrm{tt})$
(7) $\exp(\varepsilon, \underline{k}, a) = \varepsilon$
(8) $\exp(d\ \&\ s, i\ \&\ \underline{k}, a) = d^i\ \&\ \exp(s, \underline{k}, \mathrm{non}(a))$

Corollary 1.3.6.: It is $\mathrm{messt}(\exp(s, \underline{k}, a), a) = \mathrm{messt}(\exp(s, \underline{k}, \mathrm{non}(a)), \mathrm{non}(a))$.

Theorem 1.3.7.: If $s \neq \varepsilon$ then $\mathrm{messc}(c(s)) = s$ for all finite lists of data s.

<u>Proof:</u> (Induction over the length of s)
Assume for an arbitrary s $\mathrm{messc}(c(s)) = s$ holds.
There are k, k', $\underline{k}$, k'' such that $c(\mathrm{cons}(d, s)) = \mathrm{cr}^k\ \&\ \mathrm{dt}(d, \mathrm{tt})^{k'}\ \&\ \exp(s, \underline{k}, \mathrm{ff})\ \&\ \mathrm{dr}^{k''}$
Since c(s) has the form $c(s) = \mathrm{cr}^k\ \&\ \exp(s, \underline{k}, \mathrm{tt})\ \&\ \mathrm{dr}^{k''}$,
it's $\mathrm{messc}(c(s)) = \mathrm{messt}(\exp(s, \underline{k}, \mathrm{tt})\ \&\ \mathrm{dr}^{k''}, \mathrm{tt}) = s$ and
$\mathrm{messc}(c(d\ \&\ s)) = \mathrm{messt}(\mathrm{dt}(d, \mathrm{tt})^{k'}\ \&\ \exp(s, \underline{k}, \mathrm{tt})\ \&\ \mathrm{dr}^{k''}, \mathrm{tt})$
$= d\ \&\ \mathrm{messt}(\exp(s, \underline{k}, \mathrm{ff})\ \&\ \mathrm{dr}^{k''}, \mathrm{ff}) = d\ \&\ s$.

1.4. Practical experience

We proved the theorem $lf(c) \rightarrow p(c)$ given in section 1 with *Tatzelwurm* in three steps:

(1a) $lf(c) \rightarrow S_1 \wedge ... \wedge S_{16}$, using the fact that c is the least fixed point
(1b) $S_1 \wedge ... \wedge S_{16} \rightarrow S_a \wedge ... \wedge S_h$, using the fairness property of the medium
(2) $S_a \wedge ... \wedge S_h \rightarrow p(c)$

To do these proofs with *Tatzelwurm* we needed about 25 h to prove S_1, ..., S_{16}, 37 h to prove S_a, ..., S_h and 12 h to prove p. The proof of the five lemmata took 4 h.
What is not included in these times, is the time we spent on planning the proof. It took a very long time (5 months) until we found a way applying the proof arguments of the proofs (1a, b) systematically. The manner of applying these arguments is general so that we hope, that we can reduce the time spending on protocol verification closely to the above times.

1.5. Conclusion

It has been shown that fixed point theory is not only convenient to describe the behaviour of protocols, but it is also possible to find correctness proofs systematically. We saw how well-known arguments as reachability and fairness are applied during the correctness proof.
The systematic application of proof arguments promises soon to reduce verification effort drastically, so that we hope to tackle real world protocols successfully.
Our next subgoal is to implement the described method with our proof programming language.

2. Systematic Verification of in SDL specified Protocols

Abstract

SDL is a language convenient to describe communication protocols. We have to verify them formally, if they are safety critical systems. Formal verification however is a very complex task, which needs to be supported by methods and tools. We present a method how correctness proofs including safety- and liveness properties can be developed systematically. We start from a purely functional description of a communication protocol, which is mechanically derived out of the SDL description. We separate its proof into several parts and show which proof methods are convenient to tackle them. Experience with this method shows that proofs can be developed with an acceptable amount of work.

2.1. Introduction

A SDL system description contains a number of process descriptions, which are in the following called *agents*. An agent is an *extended finite automata*, which is built up from *states* and *transitions*. A state of an agent consists of one of a finite number of *agentstates* and of a *datastate*, a function σ, mapping the agents variables to their values.

There are two possibilities to prove properties of SDL systems. The first possibility is to extend modelcheckers with datastates. A contribution to this task can be found in [Di92]. As alternative we use a theorem prover, which has the advantage that we don't have to extend it in order to prove system properties expressible in first order logic. The praxis however shows that it is necessary to use a special purpose prover to do verification successfully. A prover which is designed for proofs occurring in program verification is the prover of the *Tatzelwurm* system [Käu89a,b]. It is a tableau prover [Fit90] extended by reduction procedures treating arithmetic, records and lists, datastructures typically occurring in programs. These extensions are also very helpful for protocol verification problems. However case studies showed that we can make use of the automata structures of the agents in order to define a concept which optimizes the verification process. The introduction of this concept is the objective of this paper.

We suggest to do correctness proofs in two steps. In the first step we calculate the behaviour of the system out of the behaviour of the agents. In the second step we prove the property.

In this paper we show how the first step of the proof can be done systematically. There are reflected typical tasks like reachability analysis and fairness while verifying with the presented method.

The concept bases on the formalism suggested by M. Broy [Bro87, 91], who describes systems of communicating processes with *stream processing functions*. However the work of Broy lacks an approach to its automatization, which is the contribution of this paper.

Now we want to summarize some characteristic properties and illustrations of the concept.

At first we have to transform the SDL system into a formal system, including a proof theory.

* We transform a SDL process description into a system of conditional rewrite rules.

* We transform the SDL system description into a equation system *eq*.
* The system behaviour is the solution of *eq*.
* We formalize the desired property as requirement on the solution of *eq*.

To calculate the behaviour of the SDL system we have to solve the equation system *eq*.
* We apply rewrite rules as long as there are some applicable.
* If there are no more rewrite rules applicable, we transform the equation system *eq* with its solution *c* to a equation system *eq'* with its solution *c'* and establish a relation between *c* and *c*'. Then we proceed solving *eq'*.
* The process terminates, if all the derived systems of equations are solved.

The following correspondences illustrate the verification process.
* A transition is simulated in the formal system by the application of a rewrite rule.
* A system state corresponds to a equation system and its solution.
* A relation between the solutions of the equation system corresponds to a system transition.
* All solutions of all systems of equations together with its relations can be seen as reachability graph.

* Additional assumptions about the behaviour of the system, like *Fairness* lead to a drop of solutions.

The following paper stresses the calculation of solutions of the system of system equations. The solution is a n-tupel $\{c_1, ..., c_n\}$.
We formulate the system properties by defining predicates which characterize correct solutions. For example define *P* to be the predicate, formalizing that c_i and c_j contain the same sequence of data. The proof of the validity of $P(c_i, c_j)$ is a pure predicate logic proof, which can be done with *Tatzelwurm*.

2.2. The Description of the behaviour of a SDL system

The following section describes, how a SDL system description is mapped into a formal system. The agents of a SDL system are communicating with each other by sending and receiving messages. The signals, received and sent by the agent are carrying parameters.
We call a (possibly infinite) sequence of messages a *stream*. We define some operations on streams. The function *car* takes a stream and returns its first element. The function *cdr* takes a stream and returns it without its first element. The function *&* takes either two streams and returns their concatenation or it takes a message and a stream and returns a stream, where the message is added to the input stream. We assume that ε is the empty stream.
We describe the behaviour of the agents by *stream processing functions*, mapping input streams to output streams.
Fig 2.2.1. shows a typical architecture of a communication protocol. We have two protocol

agents, two medium agents and two user agents. The streams are called $c_1, \ldots, c_8$. In 2.2.1. the behaviour of the protocol agents is described by the functions *g*, *h*, a user agent is described by a function *u*, and the medium agents are described by the medium function *med*.

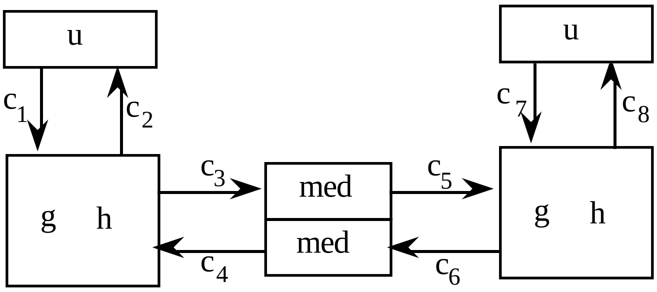

Fig. 2.2.1. A typical protocol architecture

We define the behaviour of the system to be the solution of the following equation system:
$c_1 = u(c_2)$; $c_2 = g(c_1, c_4)$; $c_3 = h(c_1, c_4)$; $c_4 = med(c_6)$;
$c_5 = med(c_3)$; $c_6 = h(c_7, c_5)$; $c_7 = u(c_8)$; $c_8 = g(c_7, c_5)$;

If we want to express properties, we do this by defining a predicate which is assumed to evaluate to *true*, if the property is fulfilled and *false* if not. For example, we define a predicate *P* with $val(P(c, c')) = true$ iff “*c* and *c*’ contain the same data”, where *val* is assumed to be an evaluation function of first order formulae.
To prove for example that $val(P(c_1, c_8)) = true$, we proceed in three steps:

1. We transform the SDL specification into a formal system, including a system of equations and rewrite rules, which are implicit definitions of the functions *u*, *g*, *h*, *med*.
2. We solve the equation system, using the derived rewrite rules.
3. We prove the required property, using the calculated solution.

In the remaining we describe how we derive a set of rewrite rules out of a SDL process description. If we analyze SDL, we realize that we have four elementary actions.
(1) Agent reads a signal.
(2) Agent evaluates a condition
(3) Agent writes a signal
(4) Agent assigns a value to a variable
In a diploma thesis [Mü93] has been shown that the SDL language constructs can be transformed into a sequence of elementary actions. Assume that the descriptions of the agents only consist of elementary actions. We regard the names of the SDL states as program labels and introduce additional labels g_i between elementary actions. The labels are mapped into 2-ary functionsymbols in the formal system. The functions (stream processing functions) take a datastate σ and an input stream *i* as their arguments and return an output stream. We define that $\sigma[s]$ means that σ is evaluated at the place *s* and that $\sigma[s \leftarrow t][s] = t$. The relation

“::=” describes a relation between the functions. A evaluation of a condition is mapped to a conditional rewrite rule, where ⇒ separates the condition from the rewrite rule. The following example shows a SDL state definition and its mapping to a system of rewrite rules.

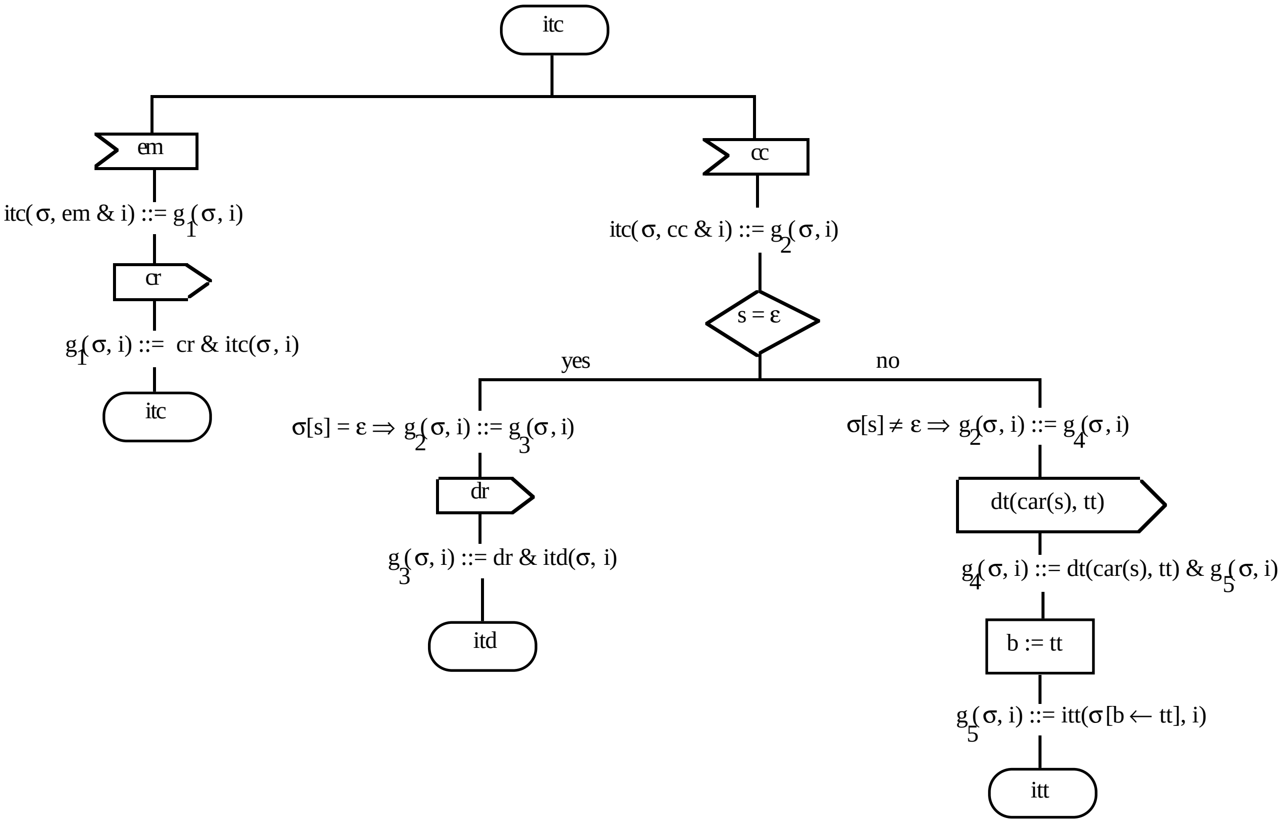
itc
em
itc(σ, em & i) ::= g1(σ, i)
cr
g1(σ, i) ::= cr & itc(σ, i)
itc
cc
itc(σ, cc & i) ::= g2(σ, i)
s = ε
yes
no
σ[s] = ε ⇒ g2(σ, i) ::= g3(σ, i)
dr
g3(σ, i) ::= dr & itd(σ, i)
itd
σ[s] ≠ ε ⇒ g2(σ, i) ::= g4(σ, i)
dt(car(s), tt)
g4(σ, i) ::= dt(car(s), tt) & g5(σ, i)
b := tt
g5(σ, i) ::= itt(σ[b ← tt], i)
itt

Fig. 2.2.2. Transformation of Agent Behaviour

In the next step we simplify the representation omitting the intermediate states, because it is inconvenient to derive such a lot of rewrite rules from the SDL text. We get for every transition path one rewrite rule, if we combine all rules situated on it. The following Fig. 2.2.3. shows the result of the transformation of the SDL fragment to rewrite rules in our formal system. Proceeding in this way we can transform SDL agent descriptions into a system of rewrite rules. The functions describing the behaviour of the system are defined implicitly by the rules, i.e. they are the models of this formal system.

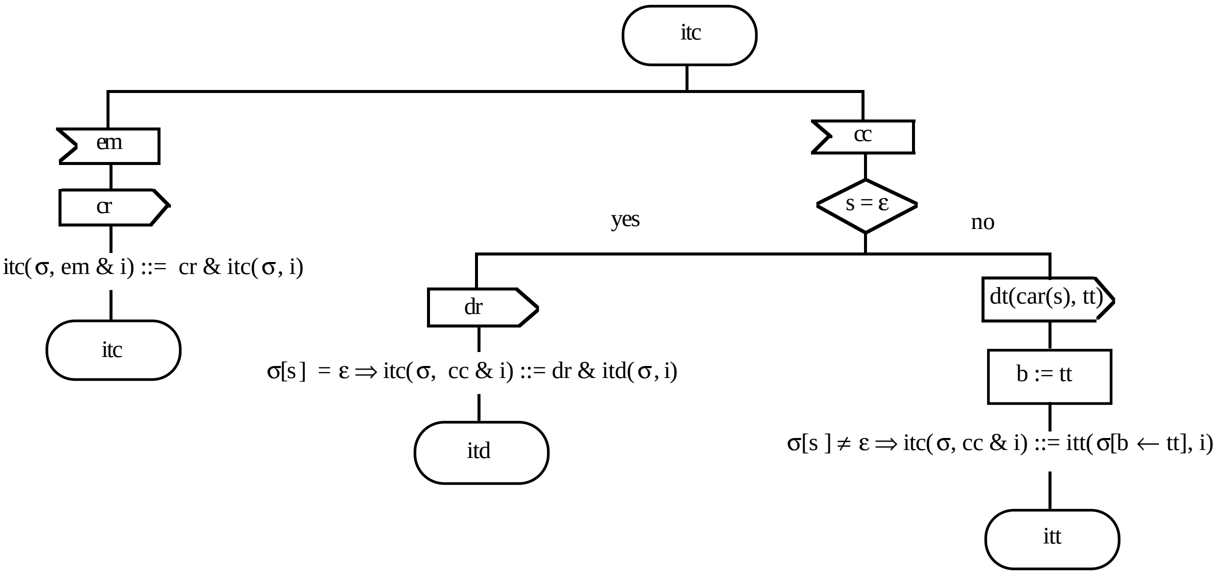
itc
em
cr
itc(σ, em & i) ::= cr & itc(σ, i)
itc
cc
s = ε
yes
no
dr
σ[s] = ε ⇒ itc(σ, cc & i) ::= dr & itd(σ, i)
itd
dt(car(s), tt)
b := tt
σ[s] ≠ ε ⇒ itc(σ, cc & i) ::= itt(σ[b ← tt], i)
itt

Fig. 2.2.3. An agent description and its mapping
to a system of rewrite rules

For further presentation we study a simplified version of the Abracadabra Protocol [Isc21]. Fig. 2.2.4. shows the architecture of this version. The protocol description is simplified in two aspects. First we use a simplified user interface. We assume that the user hands over a sequence *s* of data to the protocol. Second we assume, that we know which of the processes wants to transmit *s*. We describe the behaviour of the sender with the function *con* and the behaviour of the receiver with the function *lic*.

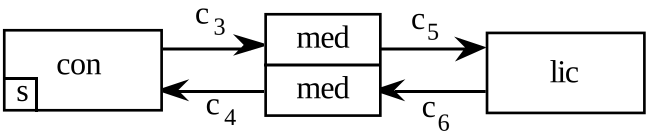

Fig. 2.2.4. A simplified version of the Abracadabra Protocol

We give now an informal description of the behaviour of the Abracadabra Protocol. The two stations communicate transferring Protocol Data Units (PDUs). Communication proceeds in the three sequential phases: *Connection*, *Data Transfer* and *Disconnection* . We assume that the *medium* is unreliable. An attempt to transmit a message *m* by a medium may succeed or fail. We model the failure of a transmission attempt by receiving a special message *em.*

a. Connection Phase

The sender sends a *cr*, a Connection Request PDU. If it receives *cc*, a Connection Confirmation PDU, it proceeds to the data transfer phase, if there are some data to transmit. If there are no data to transmit it sends *dr*, a Disconnect Request PDU and enters the disconnection phase. If it receives *em*, a *cr* is sent.

b. Data Phase

The sender sends *dt*, a Data Transfer PDU. If it receives *ak*, an acknowledgement PDU it may send a further *dt*. Each *dt* and *ak* bears a control bit, which is assumed to be *tt* or *ff*.. Successive *dt'*s carry alternating values in sequence starting at *tt* after connection. The correct acknowledgement to a *dt* contains the control bit of the next *dt* expected. The reception of an *em* or the wrong control bit causes a further attempt to transmit *dt*.

c. Disconnection Phase

The sender sends *dr*, a Disconnection Request PDU. If it receives a *dc*, a disconnection confirmation PDU, it considers the disconnection to be complete. If it receives *em*, it tries another time to transmit *dr*.

Fig. 2.2.5. shows the behaviour of the agents of the protocol using a finite state machine. Note that this is only an approximation of the behaviour of the protocol, because the datastates are omitted.

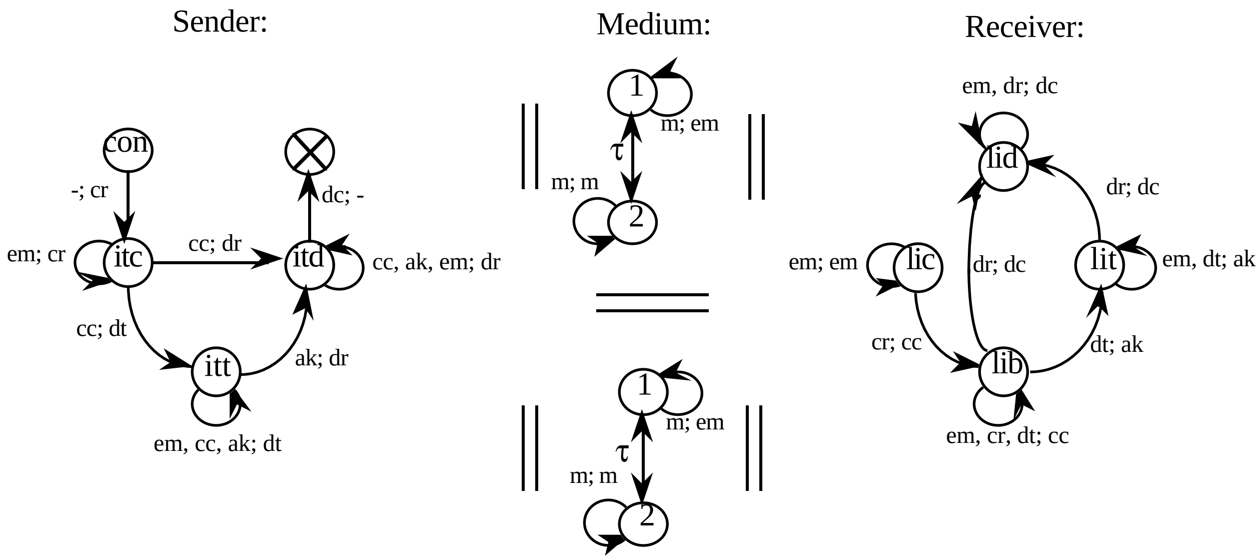

Fig. 2.2.5. The components of the Abracadabra Protocol

The rewrite rule representation, given in For. 2.2.1. describes the full behaviour of the protocol, including the datastates. The rewrite rules are constructed in the following way. They are conditioned, where *C* is assumed to be a quantor free formula, which is evaluated to *true* or *false* with respect to the actual datastate σ. *f*, *g* are function symbols, *m*, *m'* are messages and *i* is a stream.

$$C(\sigma) \Rightarrow f(\sigma, m \,\&\, i) ::= m' \,\&\, g(\sigma, i)$$

In some cases the right hand side is just ε. These rules correspond to actions, terminating a process.

Definitio[redacted]**2.2.1.:** (The behaviour of the *sender*)

Connectio[redacted]*hase*

(1) $con(i$ [redacted] $= cr \,\&\, itc(\sigma, i)$

(2) $itc(\sigma,$ [redacted] $n \,\&\, i) ::= cr \,\&\, itc(\sigma, i)$

(3) $\sigma[s] \neq$ [redacted] $\Rightarrow itc(\sigma, cc \,\&\, i) ::= dt(car(\sigma[s], tt)) \,\&\, itt(\sigma[b \leftarrow tt], i)$

(4) $\sigma[s] =$ [redacted] $\Rightarrow itc(\sigma, cc \,\&\, i) ::= dr \,\&\, itd(i)$

Data Phas[redacted]

(5) $cdr(\sigma$ [redacted] $) \neq \varepsilon \wedge \sigma[b] = v$

$\Rightarrow itt($ [redacted] $ak(v) \,\&\, i) ::= dt(car(cdr(\sigma[s])), non(v))$

$\&\, itt(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), non(v))], i)$

(6) $cdr(\sigma$ [redacted] $) = \varepsilon \wedge \sigma[b] = v \Rightarrow itt(\sigma, ak(v) \,\&\, i) ::= dr \,\&\, itd(i)$

(7) $itt(\sigma,$ [redacted] $\&\, i) ::= dt(car(\sigma[s], \sigma[b]) \,\&\, itt(\sigma, i)$

(8) $itt(\sigma,$ [redacted] $\&\, i) ::= dt(car(\sigma[s], \sigma[b]) \,\&\, itt(\sigma, i)$

(9) $\sigma[b] =$ [redacted] $\Rightarrow itt(\sigma, ak(non(v)) \,\&\, i) ::= dt(car(\sigma[s], \sigma[b]) \,\&\, itt(\sigma, i)$

Disconnec[redacted]*n Phase*

(10) $itd(dc$ [redacted] $i) ::= \varepsilon$

(11) $itd(cc$ [redacted] $i) ::= dr \,\&\, itd(i)$

(12) $itd(em$ [redacted] $i) ::= dr \,\&\, itd(i)$

(13) $itd(ak$ [redacted] $\&\, i) ::= dr \,\&\, itd(i)$

Definitio[redacted]**2.2.2.:** (The behaviour of the receiver)

Waiting fo[redacted] *connect request*

(1) $lic(cr$ [redacted] $i) ::= cc \,\&\, lib(\sigma, i)$

(2) $lic(em$ [redacted] $i) ::= em \,\&\, lic(i)$

Waiting u[redacted] *the connection is established*

(3) $lib(\sigma,$ [redacted] $(v, tt) \,\&\, i) ::= ak(tt) \,\&\, lit(\sigma[b \leftarrow ff], i)$

(4) $lib(\sigma,$ [redacted] $(v, ff) \,\&\, i) ::= cc \,\&\, lib(\sigma, i)$

(5) $lib(\sigma,$ [redacted] $\&\, i) ::= cc \,\&\, lib(\sigma, i)$

(6) $lib(\sigma,$ [redacted] $n \,\&\, i) ::= cc \,\&\, lib(\sigma, i)$

(7) $lib(\sigma,$ [redacted] $\&\, i) ::= dc \,\&\, lid(i)$

Data Trans[redacted]*r Phase*

(8) $\sigma[b] =$ [redacted] $' \Rightarrow lit(\sigma, dt(v, v') \,\&\, i) ::= ak(v') \,\&\, lit(\sigma[b \leftarrow non(v')], i)$

(9) $\sigma[b]$ [redacted] $' \Rightarrow lit(\sigma, dt(v, v') \,\&\, i) ::= ak(v') \,\&\, lit(\sigma, i)$

(10) $lit(\sigma,$ [redacted] $\&\, i) ::= ak(non(\sigma[b])) \,\&\, lit(\sigma, i)$

(11) $lit(\sigma,$ [redacted] $\&\, i) ::= dc \,\&\, lid(i)$

Waiting fo[redacted] *disconnect confirmation*

(12) $lid(em$ [redacted] $i) ::= dc \,\&\, lid(i)$

(13) $lid(dr$ [redacted] $i) ::= dc \,\&\, lid(i)$

[redacted]

we omit th[redacted] argument. Let *m* be an arbitrary signal:

(1) $med($ [redacted] $\&\, i) ::= m \,\&\, med(i)$

(2) $med($ [redacted] $\&\, i) ::= em \,\&\, med(i)$

F[redacted]. 2.2.1. The Rewrite Rule Representation of the Abracadabra Protocol

Definition 2.2.4.: (The behaviour of the simplified system)
The behaviour of the simplified system is given by the following equation system
$c_3 = con(\sigma, c_4); c_4 = med(c_6); c_5 = med(c_3); c_6 = lic(\sigma, c_5)$

As example we calculate the stream c_3. We simplify the equations of 2.2.4 to a single recursive equation, which we call *system equation*.
(F_1) $c_3 = con(\sigma, med(lic(\sigma, med(c_3))))$.

We specified the system of Fig. 2.2.4. by its system equation and a set of rewrite rules. The behaviour of the system however is given *implicitly* in terms of a recursive equation. If we want to prove that the behaviour of the system has specific properties we are forced to calculate the solution *explicitly*, which is the objective of the following sections.

2.3. Explicit behaviour

2.3.1. Solving System Equations

In order to solve the system equation we apply rewrite rules until we reach a *final situation*. A final situation has the form $c = m \,\&\, g(c)$ or $c = \varepsilon$. Having reached the final situation of the form $c = \varepsilon$, we are finished, since we have calculated the solution explicitly. If the final situation has the form $c = m \,\&\, g(c)$, we use the following theorem to define a new system equation.

Theorem 2.3.1.:
Let m be an arbitrary message and let g be a stream processing function. If c is the least solution of the equation $y = m \,\&\, g(y)$. and c' is the least solution of the equation
$y = g(m \,\&\, y)$. Then $c = m \,\&\, c'$.

Proof:
It is $c = m \,\&\, g(c)$. Note that these recursive equations are solvable due to the Fixed Point Theorem [Loe87, Ri94]. Since m is the first element of c, c can be written as c ::= m & c', where c' is a new symbol. Applying this rule to $c = m \,\&\, g(c)$, we get $m \,\&\, c' = m \,\&\, g(m \,\&\, c')$. This equation we can simplify to $c' = g(m \,\&\, c')$.

In the following subsection 2.3.1.1. we show how the system equation (F_1) is solved using the rewrite rules of For 2.2.1. and Theorem 2.3.1.

2.3.1.1 An example

1. $c_3 = con(\sigma, med(lic(\sigma', med(c_3))))$
We apply rule 2.2.1.1. and get

$c_3 = cr \ \& \ itc(\sigma, med(lic(\sigma', med(c_3))))$ (* Rule 2.2.1.1. *)

As easily can be checked in the present situation there is no further rewrite rule applicable.We know that the first element of c_3 is a connection request *cr*. Therefore there is a stream r_x and we write $c_3 = cr \ \& \ r_x$. We assume that r_x is a new symbol. With this equation we get

$cr \ \& \ r_x = cr \ \& \ itc(\sigma, med(lic(\sigma', med(cr \ \& \ r_x))))$

Now we cut the first element. This step is justified, because $cdr(l_1) = cdr(l_2)$, if $l_1 = l_2$ for all lists l_1, l_2. We get

$r_x = itc(\sigma, med(lic(\sigma', med(cr \ \& \ r_x))))$

We derived a new recursive equation applying rewrite rules and Theorem 2.3.1. We realize that we derived an equation that is different from the equation F_1. We call it F_2 and we call its solution r_0. The solution is parameterized by the internal datastates σ, σ' of the two agents, so that in principle the solution of F_2 is $r_0(\sigma, \sigma')$. However we omit the parameters, if the datastate is not changed. We describe the connection between the solutions c_3 and r_0 of the equations F_1 and F_2 by the directed equation $c_3 ::= cr \ \& \ r_0$.

As conclusion we list the results received by the above calculation.

(F_2) $r_0 = itc(\sigma, med(lic(\sigma', med(cr \ \& \ r_0))))$

(S_1) $c_3 ::= cr \ \& \ r_0$

We now proceed solving the recursive equation F_2:

2. $r_0 = itc(\sigma, med(lic(\sigma', med(cr \ \& \ r_0))))$

We notice that both rewrite rules 2.2.3.1 and as well 2.2.3.2 are applicable. This is a typical example, how indeterminism is reflected during solving recursive equations. We treat both cases and for later calculations we keep in mind which assumptions about indeterminism we made.

Case 2.1.: We assume that the Connection Request *cr* is not transmitted by the medium,
i.e. $car(med(cr \ \& \ r_0)) = em$

At first we apply five rewrite rules to get a situation of derivation, where no further rewrite rule is applicable.

$r_0 = itc(\sigma, med(lic(\sigma', med(cr \ \& \ r_0))))$

$r_0 = itc(\sigma, med(lic(\sigma', em \ \& \ med(r_0))))$ (* Rule 2.2.3.2. *)

$r_0 = itc(\sigma, med(em \ \& \ lic(\sigma', med(r_0))))$ (* Rule 2.2.2.2. *)

$r_0 = itc(\sigma, em \ \& \ med(lic(\sigma', med(r_0))))$ (* Rule 2.2.3.1. or Rule 2.2.3.2. *)

$r_0 = cr \ \& \ itc(\sigma, med(lic(\sigma', med(r_0))))$ (* Rule 2.2.1.2. *)

Applying 2.3.1. we substitute r_0 by *cr* & r_x and get the following equation by cutting *cr*.
$r_x = itc(\sigma, med(lic(\sigma', med(cr \,\&\, r_x))))$
We realize that we derived the same equation as F_2, so that we may establish the following directed equation.

(S_2) $r_0 ::= cr \,\&\, r_0$

Note, that this result is calculated under the assumption that $car(med(cr \,\&\, r_0)) = em$.We have now to treat the case for the other decision of indeterminism.

Case 2.2.: We assume that Connection Request *cr* is actually transmitted by the medium, i.e. $car(med(cr \,\&\, r_0)) = cr$

$r_0 = itc(\sigma, med(lic(\sigma', med(cr \,\&\, r_0))))$
$r_0 = itc(\sigma, med(lic(\sigma', cr \,\&\, med(r_0))))$ (* Rule 2.2.3.1 *)
$r_0 = itc(\sigma, med(cc \,\&\, lib(\sigma', med(r_0))))$ (* Rule 2.2.2.1 *)

Now we have to treat more cases. In the first case 2.2.2. the Connect Confirmation (*cc*) PDU is actually transmitted, in case 2.2.1. not.

Case 2.2.1.: We assume: $car(med(cc \,\&\, lib(\sigma', med(r_0))) = em$

$r_0 = itc(\sigma, med(cc \,\&\, \mathrm{lib}(\sigma', \mathrm{med}(\mathrm{r}_0))))$
$r_0 = itc(\sigma, em \,\&\, med(lib(\sigma', med(r_0))))$ (* Rule 2.2.3.2 *)
$r_0 = cr \,\&\, itc(\sigma, med(lib(\sigma', med(r_0))))$ (* Rule 2.2.1.2 *)

Again there is a final situation, where no rewrite rule is applicable. We use 2.3.1. and get
$r_x = itc(\sigma, med(lib(\sigma', med(cr \,\&\, r_x))))$

We notice, that this equation is none of the already calculated equations. We call its solution r_1. and define the following equation F_3 and the directed equation S_3 describing the relation between solutions.
(F_3) $r_1 = itc(\sigma, med(lib(\sigma', med(cr \,\&\, r_1))))$
(S_3) $r_0 ::= cr \,\&\, r_1$
Note that S_3 holds under the assumption that *cr* is transmitted and *cc* not.
We now treat the case that *cc* is transmitted.

Case 2.2.2.: We assume: $car(med(cc \,\&\, lib(\sigma', med(r_0))) = cc$

$r_0 = itc(\sigma, med(cc \,\&\, lib(\sigma', med(r_0))))$
$r_0 = itc(\sigma, cc \,\&\, med(lib(\sigma', med(r_0))))$ (* Rule 2.2.3.1 *)

We are now in a situation that there are two candidates applicable. Which of the candidates depends on the present datastate. We have to treat another two cases:

Case 2.2.2.1: We assume: $\sigma[s] \neq \varepsilon$

$r_0 = itc(\sigma, cc \,\&\, med(lib(\sigma', med(r_0))))$
$r_0 = dt(car(\sigma[s]), tt) \,\&\, itt(\sigma[b \leftarrow tt], med(lib(\sigma', med(r_0))))$ (* 2.2.1.3 *)
$r_x = itt(\sigma[b \leftarrow tt], med(lib(\sigma', med(dt(car(\sigma[s]), tt) \,\&\, r_x))))$

We realize that we have a *new recursive equation* F_2 with its solution r_2 and define a conditioned directed equation S_4, where the condition is reflecting the assumption about the datastate we made. S_4 also shows that the datastate is updated.

(F_4) $r_2 = itt(\sigma[b \leftarrow tt], med(lib(\sigma', med(dt(car(\sigma[s]), tt) \,\&\, r_2))))$
(S_4) $\sigma[s] \neq \varepsilon \Rightarrow r_0(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, r_2(\sigma[b \leftarrow tt], \sigma')$

Case 2.2.2.2.: We assume: $\sigma[s] = \varepsilon$

$r_0 = itc(\sigma, cc \,\&\, med(lib(\sigma', med(r_0))))$
$r_0 = dr \,\&\, itd(med(lib(\sigma, med(r_0))))$ (* 2.2.1.4 *)
$r_x = itd(med(lib(\sigma, med(dr \,\&\, r_x))))$

We define a *new recursive equation* with its solution r_3, a new symbol
(F_5) $r_3 = itd(med(lib$[redacted]$, med(dr \,\&\, r_3))))$
(S_5) $\sigma[s] = \varepsilon \Rightarrow r_0 ::$[redacted]$r \,\&\, r_3$

Procee[redacted]
the relations betw[redacted]en their solutions. In For 2.3.1. and For 2.3.2. we list the results of the complete calculati[redacted], which can be found in App II.

(F_1) $c_3 = co$[redacted]$, med(lic(\sigma', med(c_3))))$
(F_2) $r_0 = itc$[redacted]$med(lic(\sigma', med(cr \,\&\, r_0))))$
(F_3) $r_1 = itc$[redacted]$med(lib(\sigma', med(cr \,\&\, r_1))))$
(F_4) $r_2 = itt($[redacted]$med(lib(\sigma', med(dt(car(\sigma[s]), tt) \,\&\, r_2))))$
(F_5) $r_3 = itd$[redacted]$ed(lib(\sigma, med(dr \,\&\, r_3))))$
[redacted]
(F_7) $r_5 = itt$[redacted]$s, b) \leftarrow (car(s), non(v))],$
me[redacted]$it(\sigma'[b \leftarrow non(v)], med(dt(car(\sigma[s], non(v)) \,\&\, r_5))))$
(F_8) $r_6 = itd$[redacted]$ed(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_6))))$
(F_9) $r_7 = itd$[redacted]$d(lid(med(dr \,\&\, r_7))))$

For 2.3.1. The derived equations

(S$_1$) c_3 ::= c[redacted]& r_0

(S$_2$) r_0 ::= [redacted]& r_0

(S$_3$) r_0 ::= c[redacted]& r_1

(S$_4$) $\sigma[s] \neq \varepsilon$[redacted]$\Rightarrow r_0(\sigma, \sigma')$ ::= $dt(car(\sigma[s]), tt)$ & $r_2(\sigma[b \leftarrow tt], \sigma')$

(S$_5$) $\sigma[s] = \varepsilon$[redacted]$\Rightarrow r_0$ ::= dr & r_3

(S$_6$) r_1 ::= c[redacted]& r_1

(S$_7$) $\sigma[s] \neq \varepsilon$[redacted]$\Rightarrow r_1(\sigma, \sigma')$ ::= $dt(car(\sigma[s]), tt)$ & $r_2(\sigma[b \leftarrow tt], \sigma')$

(S$_8$) $\sigma[s] = \varepsilon$[redacted]$\Rightarrow r_1$ ::= dr & r_3

(S$_9$) r_2 ::= d[redacted]$ar(\sigma[s]), tt)$ & r_2

(S$_{10}$) $r_2(\sigma[b$[redacted]$tt], \sigma')$ ::= $dt(car(\sigma[s]), tt)$ & $r_4(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

(S$_{11}$) $cdr(\sigma[$[redacted]$\neq \varepsilon \Rightarrow$

r_2[redacted]$b \leftarrow v], \sigma'[b \leftarrow non(v)])$

::=[redacted]$car(\sigma[s], non(v))$ & $r_5(\sigma[(s, b) \leftarrow (cdr(s), non(v))], \sigma'[b \leftarrow non(v)])$

(S$_{12}$) $\sigma[s] =$[redacted]$\Rightarrow r_2$ ::= dr & r_6

(S$_{13}$) r_3 ::=[redacted]& r_3.

(S$_{14}$) r_3 ::=[redacted]& r_7

(S$_{15}$) r_3 ::=[redacted]

(S$_{16}$) r_4 ::=[redacted]$car(\sigma[s], non(v))$ & r_4

(S$_{17}$) $cdr(\sigma[$[redacted]$\neq \varepsilon \Rightarrow r_4(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow v])$

::= $dt(car(cdr(\sigma[s])), v)$

& $r_5(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), v)], \sigma'[b \leftarrow v])$

(S$_{18}$) $cdr(\sigma[$[redacted]$= \varepsilon \Rightarrow r_4$ ::= dr & r_6

(S$_{19}$) r_5 ::=[redacted]$car(\sigma[s]), non(v))$ & r_5

(S$_{20}$) $r_5(\sigma[b$[redacted]$non(v)], \sigma'[b \leftarrow non(v)])$

::=[redacted]$car(\sigma[s]), non(v))$ & $r_4(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow v])$

(S$_{21}$) $cdr(\sigma[$[redacted]$\neq \varepsilon \Rightarrow r_5(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow non(v)])$

::= $dt(car(\sigma[s]), v)$

& $r_5(\sigma[(s, b) \leftarrow (v, cdr(\sigma[s]))], \sigma'[b \leftarrow v])$

(S$_{22}$) $cdr(\sigma[$[redacted]$= \varepsilon \Rightarrow r_5$ ::= dr & r_6

[redacted]

(S$_{24}$) r_6 ::=[redacted]& r_7

(S$_{25}$) r_6 ::=[redacted]

(S$_{26}$) r_7 ::=[redacted]& r_7

(S$_{27}$) r_7 ::=[redacted]

For 2.3.2. The solutions and their relations

The presented method allows to calculate a stream c_3, describing the behaviour of the system. We get this result calculating successively new recursive equations and the relations between their solutions. Analyzing the results carefully we realize correspondences between the formalism of equations and their solutions on one hand and the formalism of the extended finite automata on the other hand.

We started with the systemequation $c_3 = con(\sigma, med(lic(\sigma', med(c_3))))$, which described *implicitly* the behaviour of the system. Our goal was to calculate the behaviour *explicitly*, i.e. we wanted to calculate c_3. We calculated c_3 applying rewrite rules until there was no further rule applicable.

Applying rewrite rule corresponds to a symbolic execution of a transition of the automata.

Instead we applied Theorem 2.3.1. and defined another recursive equation. These recursive equations are *implicitly* describing the behaviour of the system in a specific system state. Its solutions describes this behaviour *explicit*.

Applying Theorem 2.3.1. transforms the problem and establishes the relation between the solutions. We regard the solutions of the equations as system states and the relations between the solutions as system transitions. The set of all directed equations correspond to the systems reachability graph, so that the whole method can be seen as reachability analysis. These correspondences allow us to present the result of the calculation graphically. For presentation purposes we divide the state r_0, ..., r_7 into three phases, i.e. Connection-, Data- and Disconnect Phase. We are omitting datastates and we present a directed equation r_i ::= m & r_j by a transition between the states r_i and r_j. To make the diagram more simple we don't present the transitions between the different phases exactly.

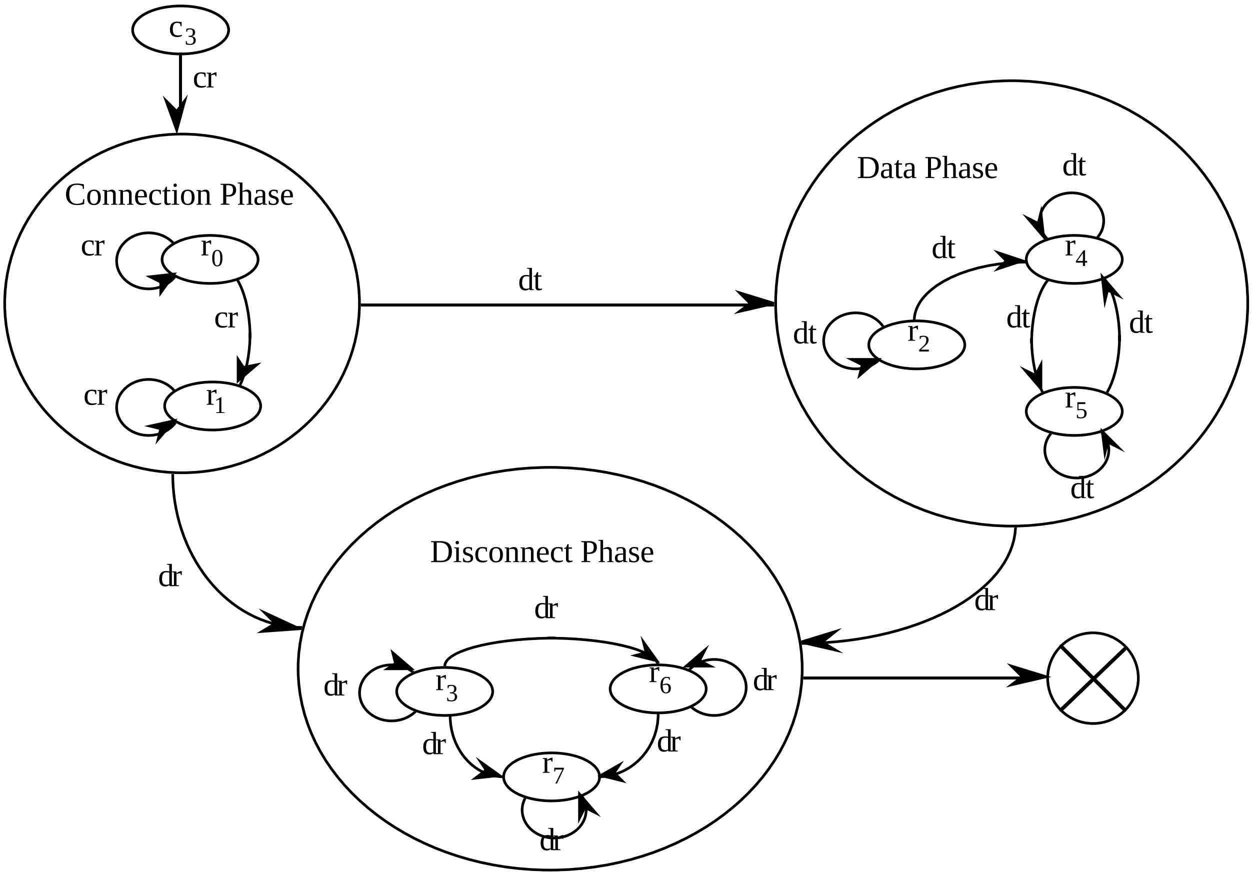

Fig. 2.3.1. The behaviour of the system

2.3.b. Liveness

After calculating the reachability graph in section 2.3.a. we have now to prove the *liveness* of the system or in other words, we have to prove that the system eventually leaves an arbitrary state after entering it. Calculating the reachability graph we said already that we have to keep in mind the assumptions we made computing the rewrite rules. Typically we have the following situations. In our system of directed equations we have pairs of rules of the following form.

$r_i ::= m \,\&\, r_i$, if $car(med(m \,\&\, g(r_i))) = em$
$r_i ::= m' \,\&\, r_j$, if $car(med(m \,\&\, g(r_i))) = m$

As example serve the rules S_2 and S_3 in our system. We have to prove that the state r_j is eventually reached. At first we prove by induction the following lemma.

Lemma: $med(m^i) = em^i \rightarrow m \,\&\, r_i = m^{i+1} \,\&\, r_i$

We prove these Lemmata by induction over *i*.
o) $i = 0$: Since $m^{i+1} = m$ the claim is trivially fulfilled
i) Let $med(m^i) = em^i \rightarrow m \,\&\, r_i = m^{i+1} \,\&\, r_i$ be true and let $med(m^{i+1}) = em^{i+1}$.
Then it is $med(m^{i+1} \,\&\, g(r_i)) = em^{i+1} \,\&\, med(g(r_i))$ and therefore
$car(med(m \,\&\, g(r_i))) = em$. We apply $r_i ::= m \,\&\, r_i$ on $m^{i+1} \,\&\, r_i$ and get
$med(m^{i+1}) = em^{i+1} \rightarrow m \,\&\, r_i = m^{i+2} \,\&\, r_i$

To proceed we give the following definition of fairness, which serves to prove that r_j is actually reached.

Definition: (Fairness)
There is a k_i such that $med(m^{k_i+1} \,\&\, r) = em^{k_i} \,\&\, m \,\&\, med(r)$

Since $med(m^{k_i}) = em^{k_i}$, we get $m \,\&\, r_i = m^{k_i+1} \,\&\, r_i$ by our last lemma
and since $car(med(m \,\&\, r_i)) = m$ we get $m \,\&\, r_i = m^{k_i+1} \,\&\, m' \,\&\, r_j$.

Proofs of the presented kind lead to a modified list of rules. For example instead of (S_2) we get (S_2') $r_0 ::= cr^{k_0} \,\&\, r_0'$ and we substitute in every left hand side of the rule each occurrence of r_0 by r_0'.

The rewrite rules in For 2.3.3. represent the *live* Connection Phase, Fig 2.3.2. is the graphical representation of this phase where the datastates are omitted.

(S$_1$) c_3 ::= [redacted] r & r_0
(S$_2´$) r_0 ::= [redacted]^{k_0} & $r_0´$
(S$_3$) $r_0´$ :: [redacted] r & r_1
[redacted]
(S$_5$) $\sigma[s]$ [redacted] ⇒ $r_0´$ ::= dr & r_3
(S$_6´$) r_1 ::= [redacted]^{k_1} & $r_1´$
(S$_7$) $\sigma[s]$ [redacted] ⇒ $r_1´(\sigma, \sigma')$ ::= $dt(car(\sigma[s]), tt)$ & $r_2(\sigma[b \leftarrow tt], \sigma')$
(S$_8$) $\sigma[s]$ [redacted] ⇒ $r_1´$ ::= dr & r_3

For 2.3.3. The *live* Connection Phase

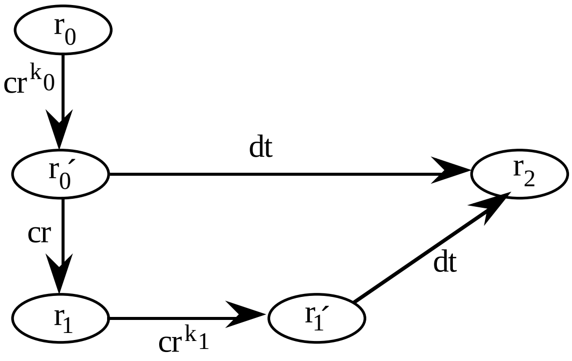

Fig. 2.3.2. The *live* Connection Phase

2.4. Evaluation

The presented paper is based on a case study, where the *Abracadabra Protocol* was verified with the program verification system *Tatzelwurm*. The verification was done in three steps and was done in a way similar to the presented method.

(1a) We calculated the the solution of the system equation. To do this we had to prove the theorems S_1, ..., S_{16}.
(1b) We proved theorems S_a, ..., S_h using the fairness property of the medium
(2) We proved the property p, we are interested in.

To do these proofs with *Tatzelwurm* we needed about 25 h to prove S_1, ..., S_{16}, 37 h to prove S_a, ..., S_h and 12 h to prove p. The proof of the five lemmata took 4 h.

What is not included in these times, is the time we spent on planning the proof. It took a very long time (5 months) until we found a way applying the proof arguments of the proofs (1a, b) systematically. The manner of applying these arguments is general so that we hope, that we can reduce the time spending on protocol verification closely to the above times.

The presented paper presented a method which allows to find correctness proofs of communication protocols systematically. It has been shown how well-known arguments as reachability and fairness are reflected during the correctness proof.

The availability of a proof procedure promises soon to reduce verification effort drastically, so that we hope to tackle real world protocols successfully.

Our next subgoal is to implement the described method with our proof programming language [De94].

3. References

[Fit90] Melvin Fitting: First Order Logic and Automated Theorem Proving, Springer 1990

[Loe87] Jacques Loeckx/ Kurt Sieber: The Foundations of Program Verification, Teubner 1987

[Bro87] Manfred Broy: Some Algebraic and Functional Hocus Pocus with Abracadabra.
MIP-8717, Fak. für Mathematik und Informatik, Universität Passau

[Bro91] Manfred Broy: Towards a Formal Foundation of the Specification and Description Language SDL
Formal Aspects of Computing 3: 21-57

[Isc21] Guidelines for the Application of Estelle, Lotos and SDL
Sec 10: Abracadabra Service and Protocol Example
ISO/IEC DTR 1067, published by ISO

[Käu89a] Th. Käufl: The Tatzelwurm User-Manual, Working Paper 1989, University of Karlsruhe

[Käu89b] Th. Käufl: Cooperation of Decision Procedures in a Tableau-Based Theorem Prover.
Technical Report 19/89, University of Karlsruhe

[Di92] Hu, Dill, Drexler, Yang: Higher-Level Specification and Verification with BDDs
Proceedings of the 4th International Workshop, CAV ‘92; LNCS 663
Springer-Verlag

[Mü93] Peter Müller: Darstellung von SDL Systemen als erweiterte Wortersetzungssysteme.
Diploma Thesis at the Univerisity Karlsruhe, 1993

[De94] Deussen P., Hansmann A., Käufl Th., Klingenbeck S.:
The Verification System *Tatzelwurm*
In Broy M., Jähnichen S. (eds.): KORSO: Correct Software by Formal Methods.
Springer LNCS (to appear)

Appendix I: The specification of the Abracadabra Protocol

As described in the presented papers we work with the following system description.

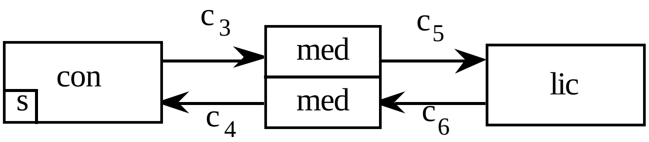

Fig. I.1. A simplified version of the Abracadabra Protocol

In the following definitions we describe the behaviour of the agents and of the system.

Definition I.1.: (The behaviour of the *sender*)

(1) $con(\sigma, i) ::= cr \;\&\; itc(\sigma, i)$

(2) $itc(\sigma, em \;\&\; i) ::= cr \;\&\; itc(\sigma, i)$

(3) $\sigma[s] \neq \varepsilon \Rightarrow itc(\sigma, cc \;\&\; i) ::= dt(car(\sigma[s], tt)) \;\&\; itt(\sigma[b \leftarrow tt], i)$

(4) $\sigma[s] = \varepsilon \Rightarrow itc(\sigma, cc \;\&\; i) ::= dr \;\&\; itd(i)$

(5) $cdr(\sigma[s]) \neq \varepsilon \wedge \sigma[b] = v$

$\Rightarrow itt(\sigma, ak(v) \;\&\; i) ::= dt(car(cdr(\sigma[s], non(v)) \;\&\; itt(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), non(v))], i)$

(6) $cdr(\sigma[s]) = \varepsilon \wedge \sigma[b] = v \Rightarrow itt(\sigma, ak(v) \;\&\; i) ::= dr \;\&\; itd(i)$

(7) $itt(\sigma, em \;\&\; i) ::= dt(car(\sigma[s], \sigma[b]) \;\&\; itt(\sigma, i)$

(8) $itt(\sigma, cc \;\&\; i) ::= dt(car(\sigma[s], \sigma[b]) \;\&\; itt(\sigma, i)$

(9) $\sigma[b] = v \Rightarrow itt(\sigma, ak(non(v)) \;\&\; i) ::= dt(car(\sigma[s], \sigma[b]) \;\&\; itt(\sigma, i)$

(10) $itd(dc \;\&\; i) ::= \varepsilon$

(11) $itd(cc \;\&\; i) ::= dr \;\&\; itd(i)$

(12) $itd(em \;\&\; i) ::= dr \;\&\; itd(i)$

(13) $itd(ak(v) \;\&\; i) ::= dr \;\&\; itd(i)$

Definition I.2.: (The behaviour of the receiver)

(1) $lic(\sigma, cr \;\&\; i) ::= cc \;\&\; lib(\sigma, i)$

(2) $lic(\sigma, em \;\&\; i) ::= em \;\&\; lic(\sigma, i)$

(3) $lib(\sigma, dt(v, tt) \;\&\; i) ::= ak(tt) \;\&\; lit(\sigma[b \leftarrow ff], i)$

(4) $lib(\sigma, dt(v, ff) \;\&\; i) ::= cc \;\&\; lib(\sigma, i)$

(5) $lib(\sigma, cr \;\&\; i) ::= cc \;\&\; lib(\sigma, i)$

(6) $lib(\sigma, em \;\&\; i) ::= cc \;\&\; lib(\sigma, i)$

(7) $lib(\sigma, dr \;\&\; i) ::= dc \;\&\; lid(i)$

(8) $\sigma[b] = v' \Rightarrow lit(\sigma, dt(v, v') \;\&\; i) ::= ak(v) \;\&\; lit(\sigma[b \leftarrow non(v')], i)$

(9) $\sigma[b] \neq v' \Rightarrow lit(\sigma, dt(v, v') \;\&\; i) ::= ak(v) \;\&\; lit(\sigma, i)$

(10) $lit(\sigma, em \;\&\; i) ::= ak(non(\sigma[b])) \;\&\; lit(\sigma, i)$

(11) $lit(\sigma, dr \& i) ::= dc \& lid(i)$

(12) $lid(em \& i) ::= dc \& lid(i)$

Definition I.3.: (The behaviour of the medium)
The behaviour of the medium doesn´t depend on a internal datastate, therefore we omit this argument. Let m be an arbitrary signal:

(1) $med(m \& i) ::= m \& med(i)$

(2) $med(m \& i) ::= em \& med(i)$

Definition I.4.: (The behaviour of the simplified system)
The behaviour of the simplified system is given by the following equation system

$c_3 = con(\sigma, c_4); c_4 = med(c_6); c_5 = med(c_3); c_6 = lic(\sigma, c_5)$

Appendix II: The complete presented example

We present the complete example of Section two. We give an explicit representation of c_3.

1. Solve $c_3 = con(\sigma, med(lic(\sigma', med(c_3))))$

(F$_1$) $c_3 = \mathrm{con}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(c_3))))$

$c_3 = \mathrm{cr} \,\&\, \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(c_3))))$ (* Rule I.1.1 *)

$\mathrm{cr} \,\&\, r_x = \mathrm{cr} \,\&\, \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_x))))$ (* $c_3 = \mathrm{cr} \,\&\, r_x$ *)

$r_x = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_x))))$

We define a *new recursive equation*, with its solution r_0, a new symbol

(F$_2$) $r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_0))))$

(S$_1$) $c_3 ::= \mathrm{cr} \,\&\, r_0$

2. Solve (F$_2$) $r_0 = itc(\sigma, med(lic(\sigma', med(cr \,\&\, r_0))))$

Case 2.1.: We assume: $car(med(cr \,\&\, r_0)) = em$

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_0))))$

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{em} \,\&\, \mathrm{med}(r_0))))$ (* Rule I.3.2 *)

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{em} \,\&\, \mathrm{lic}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.2.2 *)

$r_0 = \mathrm{itc}(\sigma, \mathrm{em} \,\&\, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.3.1 or Rule I.3.2 *)

$r_0 = \mathrm{cr} \,\&\, \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.1.2 *)

$r_x = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_x))))$, which is the *same* as (F$_2$).

We get:

(S$_2$) $r_0 ::= \mathrm{cr} \,\&\, r_0$

Case 2.2.: We assume that $car(med(cr \,\&\, r_0)) = cr$

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_0))))$

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{cr} \,\&\, \mathrm{med}(r_0))))$ (* Rule I.3.1 *)

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{cc} \,\&\, \mathrm{lib}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.2.1 *)

Case 2.2.1.: We assume: $car(med(cc \,\&\, lib(\sigma', med(r_0)))) = em$

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{cc} \,\&\, \mathrm{lib}(\sigma', \mathrm{med}(r_0))))$

$r_0 = \mathrm{itc}(\sigma, \mathrm{em} \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.3.2 *)
$r_0 = \mathrm{cr} \,\&\, \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.1.2 *)
$r_x = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_x))))$

We define a *new recursive equation*, with its solution r_1, a new symbol
(F_3) $r_1 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_1))))$ and we get
(S_3) $r_0 ::= cr \,\&\, r_1$

Case 2.2.2.: We assume: $car(med(cc \,\&\, lib(\sigma', med(r_0))) = cc$

$r_0 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{cc} \,\&\, \mathrm{lib}(\sigma', \mathrm{med}(r_0))))$
$r_0 = \mathrm{itc}(\sigma, \mathrm{cc} \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(r_0))))$ (* Rule I.3.1 *)

Case 2.2.2.1: We assume: $\sigma[s] \neq \varepsilon$

$r_0 = \mathrm{itc}(\sigma, \mathrm{cc} \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(r_0))))$
$r_0 = \mathrm{dt}(\mathrm{car}(\sigma[s], \mathrm{tt})) \,\&\, \mathrm{itt}(\sigma[b \leftarrow \mathrm{tt}], \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(r_0))))$ (* I.1.3 *)
$r_x = \mathrm{itt}(\sigma[b \leftarrow \mathrm{tt}], \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s], \mathrm{tt})) \,\&\, r_x))))$

We define a *new recursive equation* with its solution r_2, a new symbol
(F_4) $r_2 = \mathrm{itt}(\sigma[b \leftarrow \mathrm{tt}], \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{tt}) \,\&\, r_2))))$
(S_4) $\sigma[s] \neq \varepsilon \Rightarrow r_0(\sigma, \sigma') ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{tt}) \,\&\, r_2(\sigma[b \leftarrow \mathrm{tt}], \sigma')$

Case 2.2.2.2.: We assume: $\sigma[s] = \varepsilon$

$r_0 = \mathrm{itc}(\sigma, \mathrm{cc} \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(r_0))))$
$r_0 = \mathrm{dr} \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma, \mathrm{med}(r_0))))$ (* I.1.4 *)
$r_x = \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma, \mathrm{med}(\mathrm{dr} \,\&\, r_x))))$

We define a *new recursive equation* with its solution r_3, a new symbol
(F_5) $r_3 = \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma, \mathrm{med}(\mathrm{dr} \,\&\, r_3))))$
(S_5) $\sigma[s] = \varepsilon \Rightarrow r_0 ::= \mathrm{dr} \,\&\, r_3$

3. Solve (F_3): $r_1 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_1))))$

Case 3.1.: We assume $car(med(cr \,\&\, r_1)) = em$

$r_1 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{med}(\mathrm{cr} \,\&\, r_1))))$
$r_1 = \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{em} \,\&\, \mathrm{med}(r_1))))$ (* Rule I.3.2. *)

r_1 = itc(σ, med(cc & lib(σ’, med(r_1)))) (* Rule I.2.6. *)
We have now reached the same derivation as in 2.2.
So we can proceed in the same way and get the following rules:

(S_6) r_1 ::= cr & r_1
(S_7) σ[s] ≠ ε ⇒ r_1(σ, σ’) ::= dt(car(σ[s]), tt) & r_2(σ[b ← tt], σ’)
(S_8) σ[s] = ε ⇒ r_1 ::= dr & r_3

Case 3.2.: We assume *car*(*med*(*cr* & *r_1*)) = *cr*

r_1 = itc(σ, med(lib(σ’, med(cr & r_1))))
r_1 = itc(σ, med(lib(σ’, cr & med(r_1)))) (* Rule I.3.2. *)
r_1 = itc(σ, med(cc & lib(σ’, med(r_1)))) (* Rule I.2.5. *)

We have the same situation as in 3.1., so that we can derive no new equations and rules.

4. Solve (F_4): *r_2 = itt*(*σ*[*b ← tt*], *med*(*lib*(*σ*’, *med*(*dt*(*car*(*σ*[*s*]), *tt*) & *r_2*))))

Case 4.1.: We assume: *car*(*med*(*dt*(*car*(*σ*[*s*]), *tt*) & *r_2*))) = *em*

r_2 = itt(σ[b ← tt], med(lib(σ’, med(dt(car(σ[s]), tt) & r_2))))
r_2 = itt(σ[b ← tt], med(lib(σ’, em & med(r_2)))) (* Rule I.3.2. *)
r_2 = itt(σ[b ← tt], med(cc & lib(σ’, med(r_2)))) (* Rule I.2.6 *)

Case 4.1.1.: We assume: *car*(*med*(*cc* & *lib*(*σ*’, *med*(*r_2*))) = *em*

r_2 = itt(σ[b ← tt], med(cc & lib(σ’, med(r_2))))
r_2 = itt(σ[b ← tt], em & med(lib(σ’, med(r_2)))) (* Rule I.3.2. *)
r_2 = dt(car(σ[s]), tt) & itt(σ[b ← tt], med(lib(σ’, med(r_2)))) (* Rule I.1.7. *)
r_x = itt(σ[b ← tt], med(lib(σ’, med(dt(car(σ[s]), tt) & r_x))))
which ist the *same* as (F_4) and we get
(S_9) r_2 ::= dt(car(σ[s]), tt) & r_2

Case 4.1.2.: We assume: *car*(*med*(*cc* & *lib*(*σ*’, *med*(*r_1*))) = *cc*

r_2 = itt(σ[b ← tt], med(cc & lib(σ’, med(r_2))))
r_2 = itt(σ[b ← tt], cc & med(lib(σ’, med(r_2)))) (* Rule I.3.1. *)
r_2 = dt(car(σ[s]), tt) & itt(σ[b ← tt], med(lib(σ’, med(r_2)))) (* Rule I.1.8. *)

$r_x = itt(\sigma[b \leftarrow tt], med(lib(\sigma', med(dt(car(\sigma[s]), tt) \& r_x))))$,
which ist the *same* as (F_4) and we get $r_2 ::= dt(car(\sigma[s]), tt) \& r_2$, which is the *same* as (S_9).

Case 4.2.: We assume: $car(med(dt(car(\sigma[s]), tt) \& r_1))) = dt(car(\sigma[s]), tt)$

$r_2 = itt(\sigma[b \leftarrow tt], med(lib(\sigma', med(dt(car(\sigma[s]), tt) \& r_2))))$
$r_2 = itt(\sigma[b \leftarrow tt], med(lib(\sigma', dt(car(\sigma[s]), tt) \& med(r_2))))$ (* Rule I.3.1. *)
$r_2 = itt(\sigma[b \leftarrow tt], med(ak(tt) \& lit(\sigma'[b \leftarrow ff], med(r_2))))$ (* Rule I.2.3. *)

Case 4.2.1.: We assume $car(med(ak(tt) \& lit(\sigma'[b \leftarrow tt], med(r_1)))) = em$

$r_2 = itt(\sigma[b \leftarrow tt], med(ak(tt) \& lit(\sigma'[b \leftarrow ff], med(r_2))))$
$r_2 = itt(\sigma[b \leftarrow tt], em \& med(lit(\sigma'[b \leftarrow ff], med(r_2))))$ (* Rule I.3.2. *)
$r_2 = dt(car(\sigma[s]), tt) \& itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], med(r_2))))$ (* Rule I.1.7. *)
$r_x = itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], med(dt(car(\sigma[s]), tt) \& r_x))))$,

We define a *new recursive equation* with its solution r_4, a new symbol
(F_6) $r_4 = itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], med(dt(car(\sigma[s]), tt) \& r_4))))$
(S_{10}): $r_2(\sigma[b \leftarrow tt], \sigma') ::= dt(car(\sigma[s]), tt) \& r_4(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

Case 4.2.2.: We assume $car(med(ak(tt) \& lit(\sigma'[b \leftarrow ff], med(r_1)))) = ak(tt)$

$r_2 = itt(\sigma[b \leftarrow tt], med(ak(tt) \& lit(\sigma'[b \leftarrow ff], med(r_2))))$
$r_2 = itt(\sigma[b \leftarrow tt], ak(tt) \& med(lit(\sigma'[b \leftarrow ff], med(r_2))))$ (* Rule I.3.1. *)

Case 4.2.2.1.: We assume $cdr(\sigma(s)) \neq \varepsilon$

$r_2 = itt(\sigma[b \leftarrow tt], ak(tt) \& med(lit(\sigma'[b \leftarrow ff], med(r_2))))$
$r_2 = dt(car(\sigma[s], ff) \& itt(\sigma[(s, b) \leftarrow (cdr(s), ff)], ak(tt) \& med(lit(\sigma'[b \leftarrow ff], med(r_2))))$
(* Rule I.1.5 with $\tau = \{v \leftarrow tt\}$ *)

We generalize considering τ:
$r_2 = dt(car(\sigma[s], non(v)) \& itt(\sigma[(s, b) \leftarrow (cdr(s), non(v))], med(lit(\sigma'[b \leftarrow non(v)], med(r_2))))$
$r_x = itt(\sigma[(s, b) \leftarrow (cdr(s), non(v))], med(lit(\sigma'[b \leftarrow non(v)], med(dt(car(\sigma[s], non(v)) \& r_x))))$

We define a *new recursive equation* with its solution r_5, a new symbol
(F_7) $r_5 = itt(\sigma[(s, b) \leftarrow (cdr(s), non(v))], med(lit(\sigma'[b \leftarrow non(v)], med(dt(car(\sigma[s], non(v)) \& r_5))))$
(S_{11}) $cdr(\sigma[s]) \neq \varepsilon \Rightarrow$

$r_2(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$
$::= dt(car(\sigma[s], non(v)) \& r_5(\sigma[(s, b) \leftarrow (cdr(s), non(v))], \sigma'[b \leftarrow non(v)])$

Case 4.2.2.2.: We assume $cdr(s) = \varepsilon$

$r_2 = itt(\sigma[b \leftarrow tt], ak(tt) \& med(lit(\sigma'[b \leftarrow tt], med(r_2))))$
$r_2 = dr \& itd(med(lit(\sigma'[b \leftarrow tt], med(r_2))))$ (* Rule I.1.6 with $\tau = \{v \leftarrow tt\}$ *)
Considering τ we get:
$r_2 = dr \& itd(med(lit(\sigma'[b \leftarrow v], med(r_2))))$
$r_x = itd(med(lit(\sigma'[b \leftarrow v], med(dr \& r_x))))$

We define a *new generic recursive equation*, considering τ:
(F_8) $r_6 = itd(med(lit(\sigma'[b \leftarrow v], med(dr \& r_6))))$
(S_{12}) $cdr(\sigma[s]) = \varepsilon \Rightarrow r_2 = dr \& r_6$

5. Solve (F_5): $r_3 = itd(med(lib(\sigma, med(dr \& r_3))))$

Case 5.1.: $car(med(dr \& r_3)) = em$

$r_3 = itd(med(lib(\sigma, med(dr \& r_3))))$
$r_3 = itd(med(lib(\sigma, em \& med(r_3))))$ (* Rule I.3.2. *)
$r_3 = itd(med(cc \& lib(\sigma, med(r_3))))$ (* Rule I.2.6. *)

Case 5.1.1.: $car(med(cc \& lib(\sigma, med(r_3)))) = em$

$r_3 = itd(med(cc \& lib(\sigma, med(r_3))))$
$r_3 = itd(em \& med(lib(\sigma, med(r_3))))$ (* Rule I.3.2. *)
$r_3 = dr \& itd(med(lib(\sigma, med(r_3))))$ (* Rule I.1.12 *)
$r_x = itd(med(lib(\sigma, med(dr \& r_x))))$
which is the *same* as (F_5)
and we get
(S_{13}) $r_3 ::= dr \& r_3$.

Case 5.1.2.: $car(med(cc \& lib(\sigma, med(r_3)))) = cc$

$r_3 = itd(med(cc \& lib(\sigma, med(r_3))))$
$r_3 = itd(cc \& med(lib(\sigma, med(r_3))))$ (* Rule I.3.1. *)
$r_3 = dr \& itd(med(lib(\sigma, med(r_3))))$ (* Rule I.1.11. *)

$r_x = \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma, \mathrm{med}(\mathrm{dr}\ \&\ r_x))))$
which is the *same* as (F_5) and we get (S_{12})

Case 5.2.: $car(med(dr\ \&\ r_3)) = dr$

$r_3 = \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma, \mathrm{med}(\mathrm{dr}\ \&\ r_3))))$

$r_3 = \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma, \mathrm{dr}\ \&\ \mathrm{med}(r_3))))$ (* Rule I.3.1. *)

$r_3 = \mathrm{itd}(\mathrm{med}(\mathrm{dc}\ \&\ \mathrm{lid}(\mathrm{med}(r_3))))$ (* Rule I.2.7. *)

Case 5.2.1.: $car(med(dc\ \&\ lib(\sigma, med(r_3)))) = em$

$r_3 = \mathrm{itd}(\mathrm{med}(\mathrm{dc}\ \&\ \mathrm{lid}(\mathrm{med}(r_3))))$

$r_3 = \mathrm{itd}(\mathrm{em}\ \&\ \mathrm{med}(\mathrm{lid}(\mathrm{med}(r_3))))$ (* Rule I.3.2. *)

$r_3 = \mathrm{dr}\ \&\ \mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{med}(r_3))))$ (* Rule I.1.12 *)

$r_x = \mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{med}(\mathrm{dr}\ \&\ r_x))))$

We define a *new recursive equation* with its solution r_6, a new symbol

(F_8) $r_6 = \mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{med}(\mathrm{dr}\ \&\ r_6))))$

(S_{14}) $r_3 ::= \mathrm{dr}\ \&\ r_6$

Case 5.2.2.: $car(med(dc\ \&\ lib(\sigma, med(r_3)))) = dc$

$r_3 = \mathrm{itd}(\mathrm{med}(\mathrm{dc}\ \&\ \mathrm{lid}(\mathrm{med}(r_3))))$

$r_3 = \mathrm{itd}(\mathrm{dc}\ \&\ \mathrm{med}(\mathrm{lid}(\mathrm{med}(r_3))))$ (* Rule I.3.1. *)

$r_3 = \varepsilon$ (* Rule I.1.10. *)

We get

(S_{15}): $r_3 ::= \varepsilon$

6. Solve (F_6): $r_4 = \mathrm{itt}(\sigma[\mathrm{b} \leftarrow \mathrm{non}(\mathrm{v})], \mathrm{med}(\mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}], \mathrm{non}(\mathrm{v}))\ \&\ r_4))))$

Case 6.1.: $car(med(dt(car(\sigma[s], non(v))\ \&\ r_4)) = em$

$r_4 = \mathrm{itt}(\sigma[\mathrm{b} \leftarrow \mathrm{non}(\mathrm{v})], \mathrm{med}(\mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}], \mathrm{non}(\mathrm{v}))\ \&\ r_4))))$

$r_4 = \mathrm{itt}(\sigma[\mathrm{b} \leftarrow \mathrm{non}(\mathrm{v})], \mathrm{med}(\mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{em}\ \&\ \mathrm{med}(r_4))))$ (* Rule I.3.2. *)

$r_4 = \mathrm{itt}(\sigma[\mathrm{b} \leftarrow \mathrm{non}(\mathrm{v})], \mathrm{med}(\mathrm{ak}(\mathrm{non}(\mathrm{v}))\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{med}(r_4))))$ (* Rule I.2.9. *)

Case 6.1.1.: $car(med(ak(non(v))\ \&\ lit(\sigma'[b \leftarrow v], \mathrm{med}(r_4)))) = em$

$r_4 = \mathrm{itt}(\sigma[\mathrm{b} \leftarrow \mathrm{non}(\mathrm{v})], \mathrm{med}(\mathrm{ak}(\mathrm{non}(\mathrm{v}))\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{med}(r_4))))$

$r_4 = itt(\sigma[b \leftarrow non(v)], em \,\&\, med(lit(\sigma'[b \leftarrow v], med(r_4))))$ (* Rule I.3.2. *)

$r_4 = dt(car(\sigma[s]), non(v)) \,\&\, itt(\sigma[b \leftarrow non(v)], med(lit(\sigma'[b \leftarrow v], med(r_4))))$ (* Rule I.1.7. *)

$r_x = itt(\sigma[b \leftarrow non(v)], med(lit(\sigma'[b \leftarrow v], med(dt(car(\sigma[s], non(v)) \,\&\, r_x))))$

which is the same as (F_6) and we get

(S_{16}) $r_4 ::= dt(car(\sigma[s], non(v)) \,\&\, r_4$

Case 6.1.2.: $car(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], \mathrm{med}(r_4)))) = ak(non(v))$

$r_4 = itt(\sigma[b \leftarrow non(v)], med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_4))))$

$r_4 = itt(\sigma[b \leftarrow non(v)], ak(non(v)) \,\&\, med(lit(\sigma'[b \leftarrow v], med(r_4))))$ (* Rule I.3.1. *)

Case 6.1.2.1.: $cdr(\sigma[s]) \neq \varepsilon$

$r_4 = itt(\sigma[b \leftarrow non(v)], ak(non(v)) \,\&\, med(lit(\sigma'[b \leftarrow v], med(r_4))))$

$r_4 = dt(car(cdr(\sigma[s])), v) \,\&\, itt(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), v)], med(lit(\sigma'[b \leftarrow v], med(r_4))))$
(* Rule I.1.5. *)

$r_x = itt(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), v)], med(lit(\sigma'[b \leftarrow v], med(dt(car(cdr(\sigma[s])), v) \,\&\, r_x))))$

which is the same as (F_7) and we get

(S_{17}) $cdr(\sigma[s]) \neq \varepsilon \Rightarrow r_4(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow v])$
$::= dt(car(cdr(\sigma[s])), v) \,\&\, r_5(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), v)], \sigma'[b \leftarrow v])$

Case 6.1.2.2.: $cdr(\sigma[s]) = \varepsilon$

$r_4 = itt(\sigma[b \leftarrow non(v)], ak(non(v)) \,\&\, med(lit(\sigma'[b \leftarrow v], med(r_4))))$

$r_4 = dr \,\&\, itd(med(lit(\sigma'[b \leftarrow v], med(r_4))))$ (* Rule I.1.6. *)

$r_x = itd(med(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_x))))$

which is the same as (F_8) and we get

(S_{18}) $cdr(\sigma[s]) = \varepsilon \Rightarrow r_4 ::= dr \,\&\, r_6$

Case 6.2.: $car(med(dt(car(\sigma[s], non(v)) \,\&\, r_4)) = dt(car(\sigma[s], non(v))$

$r_4 = itt(\sigma[b \leftarrow non(v)], med(lit(\sigma'[b \leftarrow v], med(dt(car(\sigma[s]), non(v)) \,\&\, r_4))))$

$r_4 = itt(\sigma[b \leftarrow non(v)], med(lit(\sigma'[b \leftarrow v], dt(car(\sigma[s]), non(v)) \,\&\, med(r_4))))$ (* Rule I.3.1. *)

$r_4 = itt(\sigma[b \leftarrow non(v)], med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_4))))$

which is the same as in 6.1., so we may not hope to get something new

7. Solve (F_7): r_5 = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(dt(car(σ[s]), non(v)) & r_5))))

Case 7.1.: *car(med(dt(car(σ[s]), non(v)) & r_5) = em*

r_5 = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(dt(car(σ[s]), non(v)) & r_5))))

r_5 = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], em & med(r_5)))) (* Rule I.3.2. *)

r_5 = itt(σ[b ← non(v)], med(ak(v) & lit(σ’[b ← non(v)], med(r_5)))) (* Rule I.2.10. *)

Case 7.1.1.: *car(med(ak(v) & lit(σ’[b ← non(v)], med(r_5)))) = em*

r_5 = itt(σ[b ← non(v)], med(ak(v) & lit(σ’[b ← non(v)], med(r_5))))

r_5 = itt(σ[b ← non(v)], em & med(lit(σ’[b ← non(v)], med(r_5)))) (* Rule I.3.2. *)

r_5 = dt(car(σ[s]), non(v)) & itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(r_5))))

(* Rule I.1.7. *)

r_x = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(dt(car(σ[s]), non(v)) & r_x))))

which is the same as (F_7)

(S_{19}): r_5 ::= dt(car(σ[s]), non(v)) & r_5

Case 7.1.2.: *car(med(ak(v) & lit(σ’[b ← non(v)], med(r_5)))) = ak(v)*

r_5 = itt(σ[b ← non(v)], med(ak(v) & lit(σ’[b ← non(v)], med(r_5))))

r_5 = itt(σ[b ← non(v)], ak(v) & med(lit(σ’[b ← non(v)], med(r_5)))) (* Rule I.3.1. *)

r_5 = dt(car(σ[s]), non(v)) & itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(r_5))))

(* Rule I.1.9. *)

r_x = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(dt(car(σ[s]), non(v)) & r_x))))

which is the same as (F_7) and we get (S_{19}) r_5 ::= dt(car(σ[s]), non(v)) & r_5

Case 7.2.: *car(med(dt(car(σ[s]), non(v)) & r_5) = dt(car(σ[s]), non(v))*

r_5 = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], med(dt(car(σ[s]), non(v)) & r_5))))

r_5 = itt(σ[b ← non(v)], med(lit(σ’[b ← non(v)], dt(car(σ[s]), non(v)) & med(r_5))))

(* Rule I.3.1. *)

r_5 = itt(σ[b ← non(v)], med(ak(non(v)) & lit(σ’[b ← v], med(r_5)))) (* Rule I.2.8. *)

Case 7.2.1.: *car*(*med*(*ak*(*non*(*v*)) & *lit*(*σ*'[*b* ← *v*], *med*(r_5)))) = *em*

r_5 = itt(σ[b ← non(v)], med(ak(non(v)) & lit(σ'[b ← v], med(r_5))))
r_5 = itt(σ[b ← non(v)], em & med(lit(σ'[b ← v], med(r_5)))) (* Rule I.3.2 *)
r_5 = dt(car(σ[s], non(v)) & itt(σ[b ← non(v)], med(lit(σ'[b ← v], med(r_5))))
(* Rule I.1.7. *)
r_x = itt(σ[b ← non(v)], med(lit(σ'[b ← v], med(dt(car(σ[s], non(v)) & r_x))))

which is a specialization of (F_6). We substitute
(F_6) r_4 = itt(σ[b ← non(v)], med(lit(σ'[b ← v], med(dt(car(σ[s], non(v)) & r_4))))
(S_{20}) r_5(σ[b ← non(v)], σ'[b ← non(v)])
::= dt(car(σ[s]), non(v)) & r_4(σ[b ← non(v)], σ'[b ← v])

Case 7.2.2: *car*(*med*(*ak*(*non*(*v*)) & *lit*(*σ*'[*b* ← *v*], *med*(r_5)))) = *ak*(*non*(*v*))

r_5 = itt(σ[b ← non(v)], med(ak(non(v)) & lit(σ'[b ← v], med(r_5))))
r_5 = itt(σ[b ← non(v)], ak(non(v)) & med(lit(σ'[b ← v], med(r_5)))) (* Rule I.3.1. *)

Case 7.2.2.1.: Assume that *cdr*(*σ*[*s*]) ≠ *ε*

r_5 = itt(σ[b ← non(v)], ak(non(v)) & med(lit(σ'[b ← v], med(r_5))))
r_5 = dt(car(σ[s]), v) & itt(σ[(s, b) ← (v, cdr(σ[s]))], med(lit(σ'[b ← v], med(r_5))))
(* Rule I.1.5. *)
r_x = itt(σ[(s, b) ← (v, cdr(σ[s]))], med(lit(σ'[b ← v], med(dt(car(σ[s]), v) & r_x))))

which is the same as (F_7)
(S_{21}): cdr(σ[s]) ≠ ε ⇒ r_5(σ[b ← non(v)], σ'[b ← non(v)])
::= dt(car(σ[s]), v) & r_5(σ[(s, b) ← (v, cdr(σ[s]))], σ'[b ← v])

Case 7.2.2.2.: Assume that cdr(σ[s]) = ε

r_5 = itt(σ[b ← non(v)], ak(non(v)) & med(lit(σ'[b ← v], med(r_5))))
r_5 = dr & itd(med(lit(σ'[b ← v], med(r_5)))) (* Rule I.1.6. *)
r_x = itd(med(lit(σ'[b ← v], med(dr & r_x))))
which is the same as (F_8) and we get:
(S_{19}): cdr(σ[s]) = ε ⇒ r_5 ::= dr & r_6

8. Solve (F_8): $r_6 = itd(med(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_6))))$

Case 8.1.: Assume that $car(med(dr \,\&\, r_6)) = em$

$r_6 = itd(med(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_6))))$

$r_6 = itd(med(lit(\sigma'[b \leftarrow v], em \,\&\, med(r_6))))$ (* Rule I.3.2. *)

$r_6 = itd(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_6))))$ (* Rule I.2.10 *)

Case 8.1.1.: Assume that $car(med(ak(non(v)) \,\&\, lid(med(r_6))) = em$

$r_6 = itd(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_6))))$

$r_6 = itd(em \,\&\, med(lit(\sigma'[b \leftarrow v], med(r_6))))$ (* Rule I.3.2. *)

$r_6 = dr \,\&\, itd(med(lit(\sigma'[b \leftarrow v], med(r_6))))$ (* Rule I.1.12. *)

$r_x = itd(med(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_x))))$,

which is the same as (F_8) and we get
(S_{20}): $r_6 ::= dr \,\&\, r_6$

Case 8.1.2.: Assume that $car(med(ak(non(v)) \,\&\, lid(med(r_6))) = ak(non(v))$

$r_6 = itd(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_6))))$

$r_6 = itd(ak(non(v)) \,\&\, med(lit(\sigma'[b \leftarrow v], med(r_6))))$ (* Rule I.3.1. *)

$r_6 = dr \,\&\, itd(med(lit(\sigma'[b \leftarrow v], med(r_6))))$ (* Rule I.1.13. *)

$r_x = itd(med(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_x))))$
which is the same as (F_8) and we get (S_{19})

Case 8.2.: Assume that $car(med(dr \,\&\, r_6)) = dr$

$r_6 = itd(med(lit(\sigma'[b \leftarrow v], med(dr \,\&\, r_6))))$

$r_6 = itd(med(lit(\sigma'[b \leftarrow v], dr \,\&\, med(r_6))))$ (* Rule I.3.1. *)

$r_6 = itd(med(dc \,\&\, lid(med(r_6))))$ (* Rule I.2.11. *)

Case 8.2.1.: Assume that $car(med(dc \,\&\, lid(\mathrm{med}(r_6)))) = em$

$r_6 = itd(med(dc \,\&\, lid(med(r_6))))$

$r_6 = itd(em \,\&\, med(lid(med(r_6))))$ (* Rule I.3.2. *)

$r_6 = dr \,\&\, itd(med(lid(med(r_6))))$ (* Rule I.2.12. *)

$r_x = itd(med(lid(med(dr \,\&\, r_x))))$

which is the same as (F_8) and we get:

(S_{21}): r_6 ::= dr & r_7

Case 8.2.2.: Assume that $car(med(dc \ \& \ lid(med(r_6)))) = dc$

$r_6 = itd(med(dc \ \& \ lid(med(r_6))))$

$r_6 = itd(dc \ \& \ med(lid(med(r_6))))$ (* Rule I.3.1. *)

$r_6 = \varepsilon$ (* Rule I.1.10. *)

We get

(S_{22}): r_6 ::= ε

9. Solve (F_8): $r_7 = itd(med(lid(med(dr \ \& \ r_7))))$

Case 9.1.: Assume $car(med(dr \ \& \ r_7)) = em$

$r_7 = itd(med(lid(med(dr \ \& \ r_7)))$

$r_7 = itd(med(lid(em \ \& \ med(r_7))))$ (* Rule I.3.2. *)

$r_7 = itd(med(dc \ \& \ lid(med(r_7))))$ (* Rule I.2.11 *)

Case 9.1.1.: Assume $car(med(dc \ \& \ lid(med(r_7)))) = em$

$r_7 = itd(med(dc \ \& \ lid(med(r_7))))$

$r_7 = itd(em \ \& \ med(lid(med(r_7))))$ (* Rule I.3.2. *)

$r_7 = dr \ \& \ itd(med(lid(med(r_7))))$ (* Rule I.1.12 *)

$r_x = itd(med(lid(med(dr \ \& \ r_x))))$

which is the same as (F_9) and we get

(S_{23}): r_7 ::= dr & r_7

Case 9.1.2.: Assume $car(med(dc \ \& \ lid(med(r_7)))) = dc$

$r_7 = itd(med(dc \ \& \ lid(med(r_7))))$

$r_7 = itd(dc \ \& \ med(lid(med(r_7))))$ (* Rule I.3.1. *)

$r_7 = \varepsilon$ (* Rule I.1.10. *)

(S_{24}): r_7 ::= ε

Case 9.2.: Assume $car(med(dr \;\&\; r_7)) = dr$

$r_7 = \mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{med}(\mathrm{dr} \;\&\; r_7))))$

$r_7 = \mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{dr} \;\&\; \mathrm{med}(r_7))))$ (* Rule I.3.1. *)

$r_7 = \mathrm{itd}(\mathrm{med}(\mathrm{dc} \;\&\; \mathrm{lid}(\mathrm{med}(r_7))))$ (* Rule I.2.11. *)

We are in the same situation as in 8.1., So that we can omit the rest of the cases.

Results:

The following recursive equations have been derived:

(F_1) $c_3 = con(\sigma, med(lic(\sigma', med(c_3))))$
(F_2) $r_0 = itc(\sigma, med(lic(\sigma', med(cr \;\&\; r_0))))$
(F_3) $r_1 = itc(\sigma, med(lib(\sigma', med(cr \;\&\; r_1))))$
(F_4) $r_2 = itt(\sigma, med(lib(\sigma', med(dt(car(\sigma[s]), tt) \;\&\; r_2))))$
(F_5) $r_3 = itd(med(lib(\sigma, med(dr \;\&\; r_3))))$
(F_6) $r_4 = itt(\sigma[\mathrm{b} \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], med(dt(car(\sigma[s]), tt) \;\&\; r_4))))$
(F_7) $r_5 = itt(\sigma[(s, b) \leftarrow (cdr(s), non(v))],$
$\quad\quad med(lit(\sigma'[b \leftarrow non(v)], med(dt(car(\sigma[s], non(v)) \;\&\; r_5))))$
(F_8) $r_6 = itd(med(lit(\sigma'[b \leftarrow v], med(dr \;\&\; r_6))))$
(F_9) $r_7 = itd(med(lid(med(dr \;\&\; r_7))))$

In the following relations between the solutions together with the assumptions about indeterminisms are given:

(S_1) $c_3 ::= cr \;\&\; r_0$
$\quad\{\}$
(S_2) $r_0 ::= cr \;\&\; r_0$
$\quad\{\mathrm{car}(\mathrm{med}(\mathrm{cr} \;\&\; r_0)) = \mathrm{em}\}$
(S_3) $r_0 ::= cr \;\&\; r_1$
$\quad\{\mathrm{car}(\mathrm{med}(\mathrm{cr} \;\&\; r_0)) = \mathrm{cr}, \mathrm{car}(\mathrm{med}(\mathrm{cc} \;\&\; \mathrm{lib}(\sigma', \mathrm{med}(r_0)))) = \mathrm{em}\}$
(S_4) $\sigma[s] \neq \varepsilon \Rightarrow r_0(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \;\&\; r_2(\sigma[b \leftarrow tt], \sigma')$
$\quad\{\mathrm{car}(\mathrm{med}(\mathrm{cr} \;\&\; r_0)) = \mathrm{cr}, \mathrm{car}(\mathrm{med}(\mathrm{cc} \;\&\; \mathrm{lib}(\sigma', \mathrm{med}(r_0)))) = \mathrm{cc}\}$
(S_5) $\sigma[s] = \varepsilon \Rightarrow r_0 ::= dr \;\&\; r_3$
$\quad\{\mathrm{car}(\mathrm{med}(\mathrm{cr} \;\&\; r_0)) = \mathrm{cr}, \mathrm{car}(\mathrm{med}(\mathrm{cc} \;\&\; \mathrm{lib}(\sigma', \mathrm{med}(r_0)))) = \mathrm{cc}\}$
(S_6) $r_1 ::= cr \;\&\; r_1$
$\quad\{\mathrm{car}(\mathrm{med}(\mathrm{cr} \;\&\; r_1)) = \mathrm{cr}, \mathrm{car}(\mathrm{med}(\mathrm{cc} \;\&\; \mathrm{lib}(\sigma', \mathrm{med}(r_1)))) = \mathrm{em}\}$
(S_7) $\sigma[s] \neq \varepsilon \Rightarrow r_1(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \;\&\; r_2(\sigma[b \leftarrow tt], \sigma')$

$\{\mathrm{car}(\mathrm{med}(\mathrm{cr}\ \&\ \mathrm{r}_1)) = \mathrm{cr}, \mathrm{car}(\mathrm{med}(\mathrm{cc}\ \&\ \mathrm{lib}(\sigma', \mathrm{med}(\mathrm{r}_1))) = \mathrm{cc}\}$

(S_8) $\sigma[s] = \varepsilon \Rightarrow r_1 ::= dr\ \&\ r_3$

$\{\mathrm{car}(\mathrm{med}(\mathrm{cr}\ \&\ \mathrm{r}_1)) = \mathrm{cr}, \mathrm{car}(\mathrm{med}(\mathrm{cc}\ \&\ \mathrm{lib}(\sigma', \mathrm{med}(\mathrm{r}_1))) = \mathrm{cc}\}$

(S_9) $r_2 ::= dt(car(\sigma[s]), tt)\ \&\ r_2$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt})\ \&\ \mathrm{r}_2))) = \mathrm{em}\}$

(S_{10}) $r_2(\sigma[b \leftarrow tt], \sigma') ::= dt(car(\sigma[s]), tt)\ \&\ r_4(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt})\ \&\ \mathrm{r}_1))) = \mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt}),$

$\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{tt})\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{tt}], \mathrm{med}(\mathrm{r}_1)))) = \mathrm{em}\}$

(S_{11}) $cdr(\sigma[s]) \neq \varepsilon \Rightarrow$

$r_2(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

$::= dt(car(\sigma[s], non(v))$

$\&\ r_5(\sigma[(s, b) \leftarrow (cdr(s), non(v))], \sigma'[b \leftarrow non(v)])$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt})\ \&\ \mathrm{r}_1))) = \mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt}),$

$\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{tt})\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{ff}], \mathrm{med}(\mathrm{r}_1)))) = \mathrm{ak}(\mathrm{tt})\}$

(S_{12}) $\sigma[s] = \varepsilon \Rightarrow r_2 ::= dr\ \&\ r_6$

$\{\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt})\ \&\ \mathrm{r}_1))) = \mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}]), \mathrm{tt}),$

$\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{tt})\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{ff}], \mathrm{med}(\mathrm{r}_1)))) = \mathrm{ak}(\mathrm{tt})\}$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dr}\ \&\ \mathrm{r}_3)) = \mathrm{em},$

$\mathrm{car}(\mathrm{med}(\mathrm{cc}\ \&\ \mathrm{lib}(\sigma, \mathrm{med}(\mathrm{r}_3)))) = \mathrm{cc}\}\}$

(S_{13}) $r_3 ::= dr\ \&\ r_3.$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dr}\ \&\ \mathrm{r}_3)) = \mathrm{em},$

$\mathrm{car}(\mathrm{med}(\mathrm{cc}\ \&\ \mathrm{lib}(\sigma, \mathrm{med}(\mathrm{r}_3)))) = \mathrm{em}\}$

(S_{14}) $r_3 ::= dr\ \&\ r_6$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dr}\ \&\ \mathrm{r}_3)) = \mathrm{dr},$

$\mathrm{car}(\mathrm{med}(\mathrm{dc}\ \&\ \mathrm{lib}(\sigma, \mathrm{med}(\mathrm{r}_3)))) = \mathrm{em}\}$

(S_{15}) $r_3 ::= \varepsilon$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dr}\ \&\ \mathrm{r}_3)) = \mathrm{dr}, \mathrm{car}(\mathrm{med}(\mathrm{dc}\ \&\ \mathrm{lib}(\sigma, \mathrm{med}(\mathrm{r}_3)))) = \mathrm{dc}\}$

(S_{16}) $\mathrm{r}_4 ::= \mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}], \mathrm{non}(\mathrm{v}))\ \&\ \mathrm{r}_4$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}], \mathrm{non}(\mathrm{v}))\ \&\ \mathrm{r}_4)) = \mathrm{em},$

$\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{non}(\mathrm{v}))\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{med}(\mathrm{r}_4)))) = \mathrm{em}\}$

(S_{17}) $\mathrm{cdr}(\sigma[\mathrm{s}]) \neq \varepsilon \Rightarrow \mathrm{r}_4(\sigma[\mathrm{b} \leftarrow \mathrm{non}(\mathrm{v})], \sigma'[\mathrm{b} \leftarrow \mathrm{v}])$

$::= \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[\mathrm{s}])), \mathrm{v})\ \&\ \mathrm{r}_5(\sigma[(\mathrm{s}, \mathrm{b}) \leftarrow (\mathrm{cdr}(\sigma[\mathrm{s}]), \mathrm{v})], \sigma'[\mathrm{b} \leftarrow \mathrm{v}])$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}], \mathrm{non}(\mathrm{v}))\ \&\ \mathrm{r}_4)) = \mathrm{em},$

$\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{non}(\mathrm{v}))\ \&\ \mathrm{lit}(\sigma'[\mathrm{b} \leftarrow \mathrm{v}], \mathrm{med}(\mathrm{r}_4)))) = \mathrm{ak}(\mathrm{non}(\mathrm{v}))\}$

(S_{18}) $\mathrm{cdr}(\sigma[\mathrm{s}]) = \varepsilon \Rightarrow \mathrm{r}_4 ::= \mathrm{dr}\ \&\ \mathrm{r}_6$

$\{\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[\mathrm{s}], \mathrm{non}(\mathrm{v}))\ \&\ \mathrm{r}_4)) = \mathrm{em},$

$car(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_4)))) = ak(non(v))\}$

$(S_{19})\ r_5 ::= dt(car(\sigma[s]), non(v)) \,\&\, r_5$

$\{car(med(dt(car(\sigma[s]), non(v)) \,\&\, r_5) = em,$

$car(med(ak(v) \,\&\, lit(\sigma'[b \leftarrow non(v)], med(r_5)))) = em\}$

$(S_{20})\ r_5(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow non(v)])$

$::= dt(car(\sigma[s]), non(v)) \,\&\, r_4(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow v])$

$\{car(med(dt(car(\sigma[s]), non(v)) \,\&\, r_5) = dt(car(\sigma[s]), non(v)),$

$car(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_5)))) = em\}$

(S_{21}): $cdr(\sigma[s]) \neq \varepsilon \Rightarrow r_5(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow non(v)])$

$::= dt(car(\sigma[s]), v) \,\&\, r_5(\sigma[(s, b) \leftarrow (v, cdr(\sigma[s]))], \sigma'[b \leftarrow v])$

$\{car(med(dt(car(\sigma[s]), non(v)) \,\&\, r_5) = dt(car(\sigma[s]), non(v)),$

$car(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_5)))) = ak(non(v))\}$

(S_{22}): $cdr(\sigma[s]) = \varepsilon \Rightarrow r_5 ::= dr \,\&\, r_6$

$\{car(med(dt(car(\sigma[s]), non(v)) \,\&\, r_5) = dt(car(\sigma[s]), non(v)),$

$car(med(ak(non(v)) \,\&\, lit(\sigma'[b \leftarrow v], med(r_5)))) = ak(non(v))\}$

$(S_{23})\ r_6 ::= dr \,\&\, r_6$

$\{car(med(dr \,\&\, r_6)) = em, car(med(ak(non(v)) \,\&\, lid(med(r_6))) = em\}$

$(S_{24})\ r_6 ::= dr \,\&\, r_7$

$\{car(med(dr \,\&\, r_6)) = dr, car(med(dc \,\&\, lid(med(r_6)))) = em\}$

$(S_{25})\ r_6 ::= \varepsilon$

$\{car(med(dr \,\&\, r_6)) = dr, car(med(dc \,\&\, lid(med(r_6)))) = dc\}$

$(S_{26})\ r_7 ::= dr \,\&\, r_7$

$\{car(med(dr \,\&\, r_7)) = em, car(med(dc \,\&\, lid(med(r_7)))) = em\}$

$(S_{27})\ r_7 ::= \varepsilon$

$\{car(med(dr \,\&\, r_7)) = em, car(med(dc \,\&\, lid(med(r_7)))) = dc\}$

If we proceed as outlined in 2.3.b we get following system description including fairness.

(S_1') $c_3 ::= cr \,\&\, r_0$

(S_2') $r_0 ::= cr^{k_0} \,\&\, r_0'$

(S_3') $r_0' ::= cr \,\&\, r_1$

(S_4') $\sigma[s] \neq \varepsilon \Rightarrow r_0'(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, r_2(\sigma[b \leftarrow tt], \sigma')$

(S_5') $\sigma[s] = \varepsilon \Rightarrow r_0' ::= dr \,\&\, r_3$

(S_6') $r_1 ::= cr^{k_1} \,\&\, r_1'$

(S_7') $\sigma[s] \neq \varepsilon \Rightarrow r_1'(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, r_2(\sigma[b \leftarrow tt], \sigma')$

(S_8') $\sigma[s] = \varepsilon \Rightarrow r_1' ::= dr \,\&\, r_3$

(S_9') $r_2 ::= dt(car(\sigma[s]), tt)^{k_2} \,\&\, r_2'$

(S_{10}') $r_2'(\sigma[b \leftarrow tt], \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, r_4(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

(S$_{11}$´) $cdr(\sigma[s]) \neq \varepsilon \Rightarrow$

r_2'$(\sigma[b \leftarrow v], \sigma'[b \leftarrow non(v)])$

$::= dt(car(\sigma[s], non(v))$

$\& \; r_5(\sigma[(s, b) \leftarrow (cdr(s), non(v))], \sigma'[b \leftarrow non(v)])$

(S$_{12}$´) $\sigma[s] = \varepsilon \Rightarrow r_2' ::= dr \; \& \; r_6$

(S$_{13}$´) $r_3 ::= dr^{k_3} \; \& \; r_3'$.

(S$_{14}$´) $r_3' ::= dr \; \& \; r_7$

(S$_{15}$´) $r_3' ::= \varepsilon$

(S$_{16}$) $r_4 ::= dt(car(\sigma[s], non(v))^{k_4} \; \& \; r_4'$

(S$_{17}$) $cdr(\sigma[s]) \neq \varepsilon$

$\Rightarrow r_4'(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow v])$

$::= dt(car(cdr(\sigma[s])), v)$

$\& \; r_5(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), v)], \sigma'[b \leftarrow v])$

(S$_{17}$) $cdr(\sigma[s]) \neq \varepsilon$

(S$_{18}$) $cdr(\sigma[s]) = \varepsilon \Rightarrow r_4' ::= dr \; \& \; r_6$

(S$_{19}$) $r_5 ::= dt(car(\sigma[s]), non(v))^{k_5} \; \& \; r_5'$

(S$_{20}$) $r_5'(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow non(v)])$

$::= dt(car(\sigma[s]), non(v)) \; \& \; r_4(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow v])$

(S$_{21}$) $cdr(\sigma[s]) \neq \varepsilon$

$\Rightarrow r_5'(\sigma[b \leftarrow non(v)], \sigma'[b \leftarrow non(v)])$

$::= dt(car(\sigma[s]), v)$

$\& \; r_5'(\sigma[(s, b) \leftarrow (v, cdr(\sigma[s]))], \sigma'[b \leftarrow v])$

(S$_{22}$) $cdr(\sigma[s]) = \varepsilon \Rightarrow r_5' ::= dr \; \& \; r_6$

(S$_{23}$) $r_6 ::= dr^{k_6} \; \& \; r_6'$

(S$_{24}$) $r_6' ::= dr \; \& \; r_7$

(S$_{25}$) $r_6' ::= \varepsilon$

(S$_{26}$) $r_7 ::= dr^{k_7} \; \& \; r_7'$

(S$_{27}$) $r_7' ::= \varepsilon$

Appendix III. Another example

Again we start with the architecture of the simplified Abracadabra protocol.

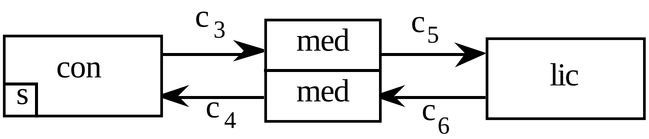

Fig. II.1. A simplified version of the Abracadabra Protocol

We now want to get closer to the problem, that the sequence of data *s* is actually received Therefore we calculate the solution of the equation c_5 = med(con(σ, med(lic(σ´, c_5)))) and show that the sequence *s* of data is contained in c_5.
We calculate c_5.

1. Solve: (F_1) c_5 = med(con(σ, med(lic(σ´, c_5))))

c_5 = med(con(σ, med(lic(σ´, c_5))))
c_5 = med(cr & itc(σ, med(lic(σ´, c_5)))) (* Rule I.1.1. *)

Case 1.1.: Assume car(med(cr & itc(σ, med(lic(c_5))))) = em

c_5 = med(cr & itc(σ, med(lic(σ´, c_5))))
c_5 = em & med(itc(σ, med(lic(σ´, c_5)))) (* Rule I.3.2. *)
r_x = med(itc(σ, med(lic(σ´, em & r_x))))

We define a *new recursive equation*, with its solution r_0, a new symbol
(F_2) r_0 = med(itc(σ, med(lic(σ´,em & r_0))))
(S_1) c_5 ::= em & r_0

Case 1.2.: Assume car(med(cr & itc(σ, med(lic(σ´, c_5))))) = cr

c_5. = med(cr & itc(σ, med(lic(σ´, c_5))))
c_5 = cr & med(itc(σ, med(lic(σ´, c_5)))) (* Rule I.3.1. *)
r_x = med(itc(σ, med(lic(σ´, cr & r_x))))

We define a *new recursive equation*, with its solution r_1, a new symbol
(F_3) r_1 = med(itc(σ, med(lic(σ´, cr & r_1))))

(S_2) c_5 ::= cr & r_1

2. Solve: (F_2) r_0 = med(itc(σ, med(lic(σ´, em & r_0))))

r_0 = med(itc(σ, med(lic(σ´, em & r_0))))
r_0 = med(itc(σ, med(em & lic(σ´, r_0)))) (* Rule I.2.2. *)
r_0 = med(itc(σ, em & med(lic(σ´, r_0)))) (* Rule I.3.1. or Rule I.3.2.*)
r_0 = med(cr & itc(σ, med(lic(σ´, r_0)))) (* Rule I.1.2. *)

Case 2.1.: Assume car(med(cr & itc(σ, med(lic(σ´, r_0))))) = em

r_0 = med(cr & itc(σ, med(lic(σ´, r_0))))
r_0 = em & med(itc(σ, med(lic(σ´, r_0)))) (* Rule I.3.2.*)
r_x = med(itc(σ, med(lic(σ´, em & r_x))))

which is the same as (F_2) and we get:
(S_3) r_0 ::= em & r_0

Case 2.2.: Assume car(med(cr & itc(σ, med(lic(σ´, r_0))))) = cr

r_0 = med(cr & itc(σ, med(lic(σ´, r_0))))
r_0 = cr & med(itc(σ, med(lic(σ´, r_0)))) (* Rule I.3.1.*)
r_x = med(itc(σ, med(lic(σ´, cr & r_x))))

which is the same as (F_3) and we get:
(S_4) r_0 ::= cr & r_1

3. Solve: (F_3) r_1 = med(itc(σ, med(lic(σ´, cr & r_1))))

r_1 = med(itc(σ, med(lic(σ´, cr & r_1))))
r_1 = med(itc(σ, med(cc & lib(σ´, r_1)))) (* Rule I.2.1. *)

Case 3.1.: Assume car(med(cc & lib(σ´, r_1))) = em

r_1 = med(itc(σ, med(cc & lib(σ´, r_1))))
r_1 = med(itc(σ, em & med(lib(σ´, r_1)))) (* Rule I.3.2. *)
r_1 = med(cr & itc(σ, med(lib(σ´, r_1)))) (* Rule I.1.2. *)

Case 3.1.1.: Assume $\mathrm{car}(\mathrm{med}(\mathrm{cr}\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))) = \mathrm{em}$

$r_1 = \mathrm{med}(\mathrm{cr}\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$

$r_1 = \mathrm{em}\ \&\ \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.2. *)

$r_x = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{em}\ \&\ r_x))))$

We define a *new recursive equation*, with its solution r_2, a new symbol

$(F_4)\ r_2 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{em}\ \&\ r_2))))$

$(S_5)\ r_1 ::= \mathrm{em}\ \&\ r_2$

Case 3.1.2.: Assume $\mathrm{car}(\mathrm{med}(\mathrm{cr}\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))) = \mathrm{cr}$

$r_1 = \mathrm{med}(\mathrm{cr}\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$

$r_1 = \mathrm{cr}\ \&\ \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.1. *)

$r_x = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{cr}\ \&\ r_x))))$

We define a *new recursive equation*, with its solution r_3, a new symbol

$(F_5)\ r_3 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{cr}\ \&\ r_3))))$

$(S_6)\ r_1 ::= \mathrm{cr}\ \&\ r_3$

Case 3.2.: Assume $\mathrm{car}(\mathrm{med}(\mathrm{cc}\ \&\ \mathrm{lib}(\sigma', r_1))) = \mathrm{cc}$

$r_1 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{cc}\ \&\ \mathrm{lib}(\sigma', r_1))))$

$r_1 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{cc}\ \&\ \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.1. *)

Case 3.2.1.: Assume $\sigma[s] \neq \varepsilon$

$r_1 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{cc}\ \&\ \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$

$r_1 = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{tt})\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.1.3. *)

Case 3.2.1.1.: Assume $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s], \mathrm{tt})\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))) = \mathrm{em}$

$r_1 = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{tt})\ \&\ \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$

$r_1 = \mathrm{em}\ \&\ \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.2. *)

$r_x = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{em}\ \&\ r_x))))$

which is the same as (F_4) and we get

$(S_7)\ \sigma[s] \neq \varepsilon \Rightarrow r_1 ::= \mathrm{em}\ \&\ r_2$

Case 3.2.1.2.: Assume $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s], tt) \,\&\, \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))) = \mathrm{dt}(\mathrm{car}(\sigma[s], tt)$

$r_1 = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), tt) \,\&\, \mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$
$r_1 = \mathrm{dt}(\mathrm{car}(\sigma[s]), tt) \,\&\, \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow tt], \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.1. *)
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow tt], \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{dt}(\mathrm{car}(\sigma[s]), tt) \,\&\, r_x))))$

We define a *new recursive equation*, with its solution r_4, a new symbol
(F_6) $r_4 = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow tt], \mathrm{med}(\mathrm{lib}(\sigma', \mathrm{dt}(\mathrm{car}(\sigma[s]), tt) \,\&\, r_4))))$
(S_8) $\sigma[s] \neq \varepsilon \Rightarrow r_1(\sigma, \sigma') ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), tt) \,\&\, \& \, r_3(\sigma[b \leftarrow tt], \sigma')$

Case 3.2.2.: Assume $\sigma[s] = \varepsilon$

$r_1 = \mathrm{med}(\mathrm{itc}(\sigma, cc \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', r_1))))$
$r_1 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.1.4. *)

Case 3.2.2.1.: Assume $\mathrm{car}(\mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))) = em$

$r_1 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))$
$r_1 = em \,\&\, \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.2. *)
$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_x))))$

We define a *new recursive equation*, with its solution r_5, a new symbol
(F_7) $r_5 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_5))))$
(S_9) $\sigma[s] = \varepsilon \Rightarrow r_1 ::= em \,\&\, r_5$

Case 3.2.2.2.: Assume $\mathrm{car}(\mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))) = dr$

$r_1 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))$
$r_1 = dr \,\&\, \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_1))))$ (* Rule I.3.1. *)
$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', dr \,\&\, r_x))))$

We define a *new recursive equation*, with its solution r_6, a new symbol
(F_8) $r_6 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', dr \,\&\, r_6))))$
(S_{10}) $\sigma[s] = \varepsilon \Rightarrow r_1 ::= dr \,\&\, r_6$

4. Solve: (F_4) $r_2 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_2))))$

$r_2 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_2))))$

r_2 = med(itc(σ, med(cc & lib(σ´, r_2)))) (* Rule I.2.6. *)

This is the same situation as in **3.**, so we get no new recursive equations, but we get the following equations.

(S_{11}) r_2 ::= em & r_2
(S_{12}) r_2 ::= cr & r_3
(S_{13}) σ[s] ≠ ε ⇒ r_2 ::= em & r_2
(S_{14}) σ[s] ≠ ε ⇒ r_2(σ, σ´) ::= dt(car(σ[s]), tt) & & r_3(σ[b ← tt], σ´)
(S_{15}) σ[s] = ε ⇒ r_2 ::= em & r_5
(S_{16}) σ[s] = ε ⇒ r_2 ::= dr & r_6

5. Solve: (F_5) r_3 = med(itc(σ, med(lib(σ´, cr & r_3))))

r_3 = med(itc(σ, med(lib(σ´, cr & r_3))))
r_3 = med(itc(σ, med(cc & lib(σ´, r_3))))

This is the same situation as in **3.**, so we get no new recursive equations, but we get the following equations.

(S_{17}) r_3 ::= em & r_2
(S_{18}) r_3 ::= cr & r_3
(S_{19}) σ[s] ≠ ε ⇒ r_3 ::= em & r_2
(S_{20}) σ[s] ≠ ε ⇒ r_3(σ, σ´) ::= dt(car(σ[s]), tt) & & r_3(σ[b ← tt], σ´)
(S_{21}) σ[s] = ε ⇒ r_3 ::= em & r_5
(S_{22}) σ[s] = ε ⇒ r_3::= dr & r_6

6. Solve: (F_6) r_4 = med(itt(σ[b ← tt], med(lib(σ´, dt(car(σ[s]), tt) & r_4))))

r_4 = med(itt(σ[b ← tt], med(lib(σ´, dt(car(σ[s]), tt) & r_4))))
r_4 = med(itt(σ[b ← tt], med(ak(tt) & lit(σ´[b ← ff], r_4)))) (* Rule I.2.3. *)

Case 6.1.: car(med(ak(tt) & lit(σ´[b ← ff], r_4))) = em

r_4 = med(itt(σ[b ← tt], med(ak(tt) & lit(σ´[b ← ff], r_4))))
r_4 = med(itt(σ[b ← tt], em & med(lit(σ´[b ← ff], r_4)))) (* Rule I.3.2. *)
r_4 = med(dt(car(σ[s]), tt) & itt(σ[b ← tt], med(lit(σ´[b ← ff], r_4)))) (* Rule I.1.7. *)

Case 6.1.1.: $car(med(dt(car(\sigma[s]), tt) \& itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], r_4))))) = em$

$r_4 = med(dt(car(\sigma[s]), tt) \& itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = em \& med(itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], r_4))))$ (* Rule I.3.2. *)

$r_x = med(itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], em \& r_x))))$

We define a *new recursive equation*, with its solution r_7, a new symbol

(F_9) $r_7 = med(itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], em \& r_7))))$

(S_{23}) $r_4(\sigma[b \leftarrow tt], \sigma') ::= em \& r_7(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

Case 6.1.2.: $car(med(dt(car(\sigma[s]), tt) \& itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], r_4)))))$
$= dt(car(\sigma[s]), tt)$

$r_4 = med(dt(car(\sigma[s]), tt) \& itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = dt(car(\sigma[s]), tt) \& med(itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], r_4))))$ (* Rule I.3.1. *)

$r_x = med(itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], dt(car(\sigma[s]), tt) \& r_x))))$

We define a *new recursive equation*, with its solution r_6, a new symbol

(F_{10}) $r_8 = med(itt(\sigma[b \leftarrow tt], med(lit(\sigma'[b \leftarrow ff], dt(car(\sigma[s]), tt) \& r_8))))$

(S_{24}) $r_4(\sigma[b \leftarrow tt], \sigma') ::= dt(car(\sigma[s]), tt) \& r_8(\sigma[b \leftarrow tt], \sigma'[b \leftarrow ff])$

Case 6.2.: $car(med(ak(tt) \& lit(\sigma'[b \leftarrow ff], r_4))) = ak(tt)$

$r_4 = med(itt(\sigma[b \leftarrow tt], med(ak(tt) \& lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = med(itt(\sigma[b \leftarrow tt], ak(tt) \& med(lit(\sigma'[b \leftarrow ff], r_4))))$ (* Rule I.3.1. *)

Case 6.2.1.: Assume $cdr(\sigma[s]) \neq \varepsilon$

$r_4 = med(itt(\sigma[b \leftarrow tt], ak(tt) \& med(lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = med(dt(car(cdr(\sigma[s])), non(v)) \& itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4))))$

(* Rule I.1.5. with $\{v \leftarrow tt\}$ *)

Case 6.2.1.1.: $car(med(dt(car(cdr(\sigma[s])), non(v))$
$\& itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4))))) = em$

$r_4 = med(dt(car(cdr(\sigma[s])), non(v)) \& itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4))))$

$r_4 = em \& med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4))))$ (* Rule I.3.2. *)

$r_x = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], em \& r_x))))$

This equation is generalization of (F_9). We substitute (F_9) by

$(F_9)\ r_7 = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], em \ \& \ r_7))))$

$(S_{25})\ cdr(\sigma[s]) \neq \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= em \ \& \ r_7(\sigma[b \leftarrow v], \sigma'[b \leftarrow non(v)])$

Case 6.2.1.2.: $car(med(dt(car(cdr(\sigma[s])), non(v))$
$\qquad \& \ itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4)))))$
$\quad = dt(car(cdr(\sigma[s])), non(v))$

$r_4 = med(dt(car(cdr(\sigma[s])), non(v)) \ \& \ itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4))))$

$r_4 = dt(car(cdr(\sigma[s])), non(v)) \ \& \ med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_4))))$

(* Rule I.3.1. *)

$r_x = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], dt(car(cdr(\sigma[s])), non(v)) \ \& \ r_x))))$

This equation is generalization of (F_{10}). We substitute (F_{10}) by

$(F_{10})\ r_8 = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], dt(car(\sigma[s]), v) \ \& \ r_8))))$

$(S_{26})\ cdr(\sigma[s]) \neq \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= dt(car(\sigma[s]), v) \ \& \ r_8(\sigma[b \leftarrow v], \sigma'[b \leftarrow non(v)])$

Case 6.2.2.: Assume $cdr(\sigma[s]) = \varepsilon$

$r_4 = med(itt(\sigma[b \leftarrow tt], ak(tt) \ \& \ med(lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = med(dr \ \& \ itd(med(lit(\sigma'[b \leftarrow ff], r_4))))$ (* Rule I.1.4. *)

Case 6.2.2.1.: $car(med(dr \ \& \ itd(med(lit(\sigma'[b \leftarrow ff], r_4))))) = em$

$r_4 = med(dr \ \& \ itd(med(lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = em \ \& \ med(itd(med(lit(\sigma'[b \leftarrow ff], r_4))))$ (* Rule I.3.2. *)

$r_x = med(itd(med(lit(\sigma'[b \leftarrow ff], em \ \& \ r_x))))$

We define a *new recursive equation*, with its solution r_9, a new symbol

$(F_{11})\ r_9 = med(itd(med(lit(\sigma'[b \leftarrow ff], em \ \& \ r_9))))$

$(S_{27})\ cdr(\sigma[s]) = \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= em \ \& \ r_9(\sigma'[b \leftarrow ff])$

Case 6.2.2.2.: $car(med(dr \ \& \ itd(med(lit(\sigma'[b \leftarrow ff], r_4))))) = dr$

$r_4 = med(dr \ \& \ itd(med(lit(\sigma'[b \leftarrow ff], r_4))))$

$r_4 = dr \ \& \ med(itd(med(lit(\sigma'[b \leftarrow ff], r_4))))$ (* Rule I.3.1. *)

$r_x = med(itd(med(lit(\sigma'[b \leftarrow ff], dr \ \& \ r_x))))$

We define a *new recursive equation*, with its solution r_{10}, a new symbol

(F_{12}) $r_{10} = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow ff], dr \,\&\, r_{10}))))$

(S_{28}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= dr \,\&\, r_{10}(\sigma'[b \leftarrow ff])$

7. Solve: (F_7) $r_5 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_5))))$

$r_5 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_5))))$

$r_5 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(cc \,\&\, \mathrm{lib}(\sigma', r_5))))$ (* Rule I.2.6. *)

Case 7.1.: $\mathrm{car}(\mathrm{med}(cc \,\&\, \mathrm{lib}(\sigma', r_5))) = em$

$r_5 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(cc \,\&\, \mathrm{lib}(\sigma', r_5))))$

$r_5 = \mathrm{med}(\mathrm{itd}(em \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', r_5))))$ (* Rule I.3.2. *)

$r_5 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))$ (* Rule I.1.12. *)

Case 7.1.1.: $\mathrm{car}(\mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))) = em$

$r_5 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))$

$r_5 = em \,\&\, \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))$ (* Rule I.3.2. *)

$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', em \,\&\, r_x))))$

The recursive equation is the same as (F_7)

(S_{29}) $r_5 ::= em \,\&\, r_5$

Case 7.1.2.: $\mathrm{car}(\mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))) = dr$

$r_5 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))$

$r_5 = dr \,\&\, \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))$ (* Rule I.3.1. *)

$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', dr \,\&\, r_x))))$

The recursive equation is the same as (F_8)

(S_{30}) $r_5 ::= dr \,\&\, r_6$

Case 7.2.: $\mathrm{car}(\mathrm{med}(cc \,\&\, \mathrm{lib}(\sigma', r_5))) = cc$

$r_5 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(cc \,\&\, \mathrm{lib}(\sigma', r_5))))$

$r_5 = \mathrm{med}(\mathrm{itd}(cc \,\&\, \mathrm{med}(\mathrm{lib}(\sigma', r_5))))$ (* Rule I.3.1. *)

$r_5 = \mathrm{med}(dr \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lib}(\sigma', r_5))))$ (* Rule I.1.11. *)

Case 7.2.1.: car(med(dr & itd(med(lib(σ', r_5))))) = em

r_5 = med(dr & itd(med(lib(σ', r_5))))
r_5 = em & med(itd(med(lib(σ', r_5)))) (* Rule I.3.2. *)
r_x = med(itd(med(lib(σ', em & r_x))))

The recursive equation is the same as (F_7) and we get (S_{29})

Case 7.2.2.: car(med(dr & itd(med(lib(σ', r_5))))) = dr

r_5 = med(dr & itd(med(lib(σ', r_5))))
r_5 = dr & med(itd(med(lib(σ', r_5)))) (* Rule I.3.1. *)
r_x = med(itd(med(lib(σ', dr & r_x))))

The recursive equation is the same as (F_8) and we get (S_{30})

8. Solve: (F_8) r_6 = med(itd(med(lib(σ', dr & r_6))))

r_6 = med(itd(med(lib(σ', dr & r_6))))
r_6 = med(itd(med(dc & lid(r_6)))) (* Rule I.2.7. *)

Case 8.1.: car(med(dc & lid(r_6))) = em

r_6 = med(itd(med(dc & lid(r_6))))
r_6 = med(itd(em & med(lid(r_6)))) (* Rule I.3.2. *)
r_6 = med(dr & itd(med(lid(r_6)))) (* Rule I.1.12 *)

Case 8.1.1.: car(med(dr & itd(med(lid(r_6))))) = em

r_6 = med(dr & itd(med(lid(r_6))))
r_6 = em & med(itd(med(lid(r_6)))) (* Rule I.3.2. *)
r_x = med(itd(med(lid(em & r_x))))

We define a *new recursive equation*, with its solution r_{11}, a new symbol
(F_{13}) r_{11} = med(itd(med(lid(em & r_{11}))))
(S_{31}) r_6 ::= em & r_{11}

Case 8.1.2.: $\mathrm{car(med(dr\ \&\ itd(med(lid(r_6))))) = dr}$

$\mathrm{r_6 = med(dr\ \&\ itd(med(lid(r_6))))}$
$\mathrm{r_6 = dr\ \&\ med(itd(med(lid(r_6))))}$ (* Rule I.3.1. *)
$\mathrm{r_x = med(itd(med(lid(dr\ \&\ r_x))))}$

We define a *new recursive equation*, with its solution r_{12}, a new symbol
(F_{14}) $\mathrm{r_{12} = med(itd(med(lid(dr\ \&\ r_{12}))))}$
(S_{32}) $\mathrm{r_6 ::= dr\ \&\ r_{12}}$

Case 8.2.: $\mathrm{car(med(dc\ \&\ lid(r_6))) = dc}$

$\mathrm{r_6 = med(itd(med(dc\ \&\ lid(r_6))))}$
$\mathrm{r_6 = med(itd(dc\ \&\ med(lid(r_6))))}$ (* Rule I.3.1. *)
$\mathrm{r_6 = med(\varepsilon) = \varepsilon}$ (* Rule I.1.10. *)

(S_{33}) $\mathrm{r_6 ::= \varepsilon}$

9. Solve: (F_9) $\mathrm{r_7 = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], em\ \&\ r_7))))}$

$\mathrm{r_7 = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], em\ \&\ r_7))))}$
$\mathrm{r_7 = med(itt(\sigma[b \leftarrow v], med(ak(v)\ \&\ lit(\sigma'[b \leftarrow non(v)], r_7))))}$ (* Rule I.2.10. *)

Case 9.1.: $\mathrm{car(med(ak(v)\ \&\ lit(\sigma'[b \leftarrow non(v)], r_7))) = em}$

$\mathrm{r_7 = med(itt(\sigma[b \leftarrow v], med(ak(v)\ \&\ lit(\sigma'[b \leftarrow non(v)], r_7))))}$
$\mathrm{r_7 = med(itt(\sigma[b \leftarrow v], em\ \&\ med(lit(\sigma'[b \leftarrow non(v)], r_7))))}$ (* Rule I.3.2. *)
$\mathrm{r_7 = med(dt(car(\sigma[s]), v)\ \&\ itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_7))))}$ (* Rule I.1.7. *)

Case 9.1.1.: $\mathrm{car(med(dt(car(\sigma[s]), v)\ \&\ itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_7)))))}$
$\mathrm{= em}$

$\mathrm{r_7 = med(dt(car(\sigma[s]), v)\ \&\ itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_7))))}$
$\mathrm{r_7 = em\ \&\ med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], r_7))))}$ (* Rule I.3.2. *)
$\mathrm{r_x = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], em\ \&\ r_x))))}$

The recursive equation is the same as (F_9) and we get
(S_{34}) $\mathrm{r_7 ::= em\ \&\ r_7}$

Case 9.1.2.: $\text{car}(\text{med}(\text{dt}(\text{car}(\sigma[s]), v)\ \&\ \text{itt}(\sigma[b \leftarrow v], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7)))))$
$= \text{dt}(\text{car}(\sigma[s]), v)$

$r_7 = \text{med}(\text{dt}(\text{car}(\sigma[s]), v)\ \&\ \text{itt}(\sigma[b \leftarrow v], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$
$r_7 = \text{dt}(\text{car}(\sigma[s]), v)\ \&\ \text{med}(\text{itt}(\sigma[b \leftarrow v], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$ (* Rule I.3.1. *)
$r_x = \text{med}(\text{itt}(\sigma[b \leftarrow v], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], \text{dt}(\text{car}(\sigma[s]), v)\ \&\ r_x))))$

The recursive equation is the same as (F_{10}) and we get
(S_{35}) $r_7 ::= \text{dt}(\text{car}(\sigma[s]), v)\ \&\ r_8$

Case 9.2.: $\text{car}(\text{med}(\text{ak}(v)\ \&\ \text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))) = \text{ak}(v)$

$r_7 = \text{med}(\text{itt}(\sigma[b \leftarrow v], \text{med}(\text{ak}(v)\ \&\ \text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$
$r_7 = \text{med}(\text{itt}(\sigma[b \leftarrow v], \text{ak}(v)\ \&\ \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$ (* Rule I.3.1. *)

Case 9.2.1.: $\text{cdr}(\sigma[s]) \neq \varepsilon$

$r_7 = \text{med}(\text{itt}(\sigma[b \leftarrow v], \text{ak}(v)\ \&\ \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$
$r_7 = \text{med}(\text{dt}(\text{car}(\text{cdr}(\sigma[s])), \text{non}(v))\ \&$
$\text{itt}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$ (* Rule I.1.5. *)

Case 9.2.1.1.: $\text{car}(\text{med}(\text{dt}(\text{car}(\text{cdr}(\sigma[s])), \text{non}(v))\ \&$
$\text{itt}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))))$
$= \text{em}$

$r_7 = \text{med}(\text{dt}(\text{car}(\text{cdr}(\sigma[s])), \text{non}(v))\ \&$
$\text{itt}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$
$r_7 = \text{em}\ \&\ \text{med}(\text{itt}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], r_7))))$
(* Rule I.3.2. *)
$r_x = \text{med}(\text{itt}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], \text{em}\ \&\ r_x))))$

We define a *new recursive equation*, with its solution r_{13}, a new symbol
(F_{15}) $r_{13} = \text{med}(\text{itt}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \text{med}(\text{lit}(\sigma'[b \leftarrow \text{non}(v)], \text{em}\ \&\ r_{13}))))$
(S_{36}) $\text{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow$
$r_7(\sigma[b \leftarrow v], \sigma'[b \leftarrow \text{non}(v)])$
$::= \text{em}\ \&\ r_{13}(\sigma[(s, b) \leftarrow (\text{cdr}(\sigma[s]), \text{non}(v))], \sigma'[b \leftarrow \text{non}(v)])$

Case 9.2.1.2.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), \mathrm{non}(v)) \,\&$
$\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))))$
$= \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), \mathrm{non}(v))$

$r_7 = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), \mathrm{non}(v)) \,\&$
$\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))$
$r_7 = \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), \mathrm{non}(v))$
$\&\, \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))$
(* Rule I.3.1. *)
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],$
$\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), \mathrm{non}(v)) \,\&\, r_x))))$

We define a *new recursive equation*, with its solution r_{14}, a new symbol
(F_{16}) $r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],$
$\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), \mathrm{non}(v)) \,\&\, r_{14}))))$
(S_{37}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow$
$r_7(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$
$::= \mathrm{em} \,\&\, r_{14}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \sigma'[b \leftarrow \mathrm{non}(v)])$

Case 9.2.2.: $\mathrm{cdr}(\sigma[s]) = \varepsilon$

$r_7 = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow v], \mathrm{ak}(v) \,\&\, \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))$
$r_7 = \mathrm{med}(\mathrm{dr} \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))$ (* Rule I.1.6. *)

Case 9.2.2.1.: $\mathrm{car}(\mathrm{med}(\mathrm{dr} \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))) = \mathrm{em}$

$r_7 = \mathrm{med}(\mathrm{dr} \,\&\, \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))$
$r_7 = \mathrm{em} \,\&\, \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_7))))$ (* Rule I.3.2. *)
$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], \mathrm{em} \,\&\, r_x))))$

The recursive equation is a specialization of (F_{11}). We substitute:
(F_{11}) $r_9 = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], \mathrm{em} \,\&\, r_9))))$
(S_{38}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_7 ::= \mathrm{em} \,\&\, r_9$

Case 9.2.2.2.: car(med(dr & itd(med(lit(σ´[b ← non(v)], r_7))))) = dr

r_7 = med(dr & itd(med(lit(σ´[b ← non(v)], r_7))))
r_7 = dr & med(itd(med(lit(σ´[b ← non(v)], r_7)))) (* Rule I.3.1. *)
r_x = med(itd(med(lit(σ´[b ← non(v)], dr & r_x))))

The recursive equation is a specialization of (F_{12}). We substitute:
(F_{12}) r_{10} = med(itd(med(lit(σ´[b ← non(v)], dr & r_{10}))))
(S_{39}) cdr(σ[s]) = ε ⇒ r_7 ::= dr & r_{10}

10. Solve: (F_{10}) r_8 = med(itt(σ[b ← v], med(lit(σ´[b ← non(v)], dt(car(σ[s]), v) & r_8))))

r_8 = med(itt(σ[b ← v], med(lit(σ´[b ← non(v)], dt(car(σ[s]), v) & r_8))))
r_8 = med(itt(σ[b ← v], med(ak(v) & lit(σ´[b ← non(v)] & r_8))))

We can now proceed as in **9.** and get the following results

(S_{40}) r_8 ::= em & r_7
(S_{41}) r_8 ::= dt(car(σ[s]), v) & r_8
(S_{42}) cdr(σ[s]) ≠ ε ⇒
r_8(σ[b ← v], σ´[b ← non(v)])
::= em & r_{13}(σ[(s, b) ← (cdr(σ[s]), non(v))], σ´[b ← non(v)])
(S_{43}) cdr(σ[s]) ≠ ε ⇒
r_8(σ[b ← v], σ´[b ← non(v)])
::= em & r_{14}(σ[(s, b) ← (cdr(σ[s]), non(v))], σ´[b ← non(v)])
(S_{44}) cdr(σ[s]) = ε ⇒ r_8 ::= em & r_9
(S_{45}) cdr(σ[s]) = ε ⇒ r_8 ::= dr & r_{10}

11. Solve: (F_{11}) r_9 = med(itd(med(lit(σ´[b ← non(v)], em & r_9))))

r_9 = med(itd(med(lit(σ´[b ← non(v)], em & r_9))))
r_9 = med(itd(med(ak(v) & lit(σ´[b ← non(v)], r_9)))) (* Rule I.2.10. *)

Case 11.1.: car(med(ak(v) & lit(σ´[b ← non(v)], r_9))) = em

r_9 = med(itd(med(ak(v) & lit(σ´[b ← non(v)], r_9))))
r_9 = med(itd(em & med(lit(σ´[b ← non(v)], r_9)))) (* Rule I.3.2. *)

r_9 = med(dr & itd(med(lit(σ´[b ← non(v)], r_9)))) (* Rule I.1.12. *)

Case 11.1.1.: car(med(dr & itd(med(lit(σ´[b ← non(v)], r_9))))) = em

r_9 = med(dr & itd(med(lit(σ´[b ← non(v)], r_9))))
r_9 = em & med(itd(med(lit(σ´[b ← non(v)], r_9)))) (* Rule I.3.2. *)
r_x = med(itd(med(lit(σ´[b ← non(v)], em & r_x))))

The recursive equation is the same as (F_{11})
(S_{46}) r_9 ::= em & r_9

Case 11.1.2.: car(med(dr & itd(med(lit(σ´[b ← non(v)], r_9))))) = dr

r_9 = med(dr & itd(med(lit(σ´[b ← non(v)], r_9))))
r_9 = dr & med(itd(med(lit(σ´[b ← non(v)], r_9)))) (* Rule I.3.2. *)
r_x = med(itd(med(lit(σ´[b ← non(v)], dr & r_x))))

The recursive equation is the same as (F_{12})
(S_{47}) r_9 ::= em & r_{10}

Case 11.2.: car(med(ak(v) & lit(σ´[b ← non(v)], r_9))) = ak(v)

r_9 = med(itd(med(ak(v) & lit(σ´[b ← non(v)], r_9))))
r_9 = med(itd(ak(v) & med(lit(σ´[b ← non(v)], r_9)))) (* Rule I.3.1. *)
r_9 = med(dr & itd(med(lit(σ´[b ← non(v)], r_9))))

We are in the same situation as in **11.1.**

12. Solve: (F_{12}) r_{10} = med(itd(med(lit(σ´[b ← non(v)], dr & r_{10}))))

r_{10} = med(itd(med(lit(σ´[b ← non(v)], dr & r_{10}))))
r_{10} = med(itd(med(dc & lid(r_{10}))) (* Rule I.2.11. *)

Case 12.1.: car(med(dc & lid(r_{10}))) = em

r_{10} = med(itd(med(dc & lid(r_{10})))
r_{10} = med(itd(em & med(lid(r_{10}))) (* Rule I.3.2. *)
r_{10} = med(dr & itd(med(lid(r_{10}))) (* Rule I.1.12 *)

Case 12.1.1: car(med(dr & itd(med(lid(r_{10})))) = em

r_{10} = med(dr & itd(med(lid(r_{10})))
r_{10} = em & med(itd(med(lid(r_{10}))) (* Rule I.3.2. *)
r_x = med(itd(med(lid(em & r_x)))

The recursive equation is the same as (F_{13}) with its solution r_{11}.
(S_{48}) r_{10} ::= em & r_{11}

Case 12.1.2.: car(med(dr & itd(med(lid(r_{10})))) = dr

r_{10} = med(dr & itd(med(lid(r_{10})))
r_{10} = dr & med(itd(med(lid(r_{10}))) (* Rule I.3.1. *)
r_x = med(itd(med(lid(dr & r_x)))

The recursive equation is the same as (F_{14}) with its solution r_{12}.

(S_{49}) r_{10} ::= dr & r_{12}

Case 12.2.: car(med(dc & lid(r_{10}))) = dc

r_{10} = med(itd(med(dc & lid(r_{10})))
r_{10} = med(itd(dc & med(lid(r_{10})))
r_{10} = med(ε) = ε (* Rule I.1.10. *)

(S_{50}) r_{10} ::= ε

13. Solve: (F_{13}) r_{11} = med(itd(med(lid(em & r_{11}))))

r_{11} = med(itd(med(lid(em & r_{11}))))
r_{11} = med(itd(med(dc & lid(r_{11})))) (* Rule I.2.12. *)

We are in the same situation as in **12.**, we get:

(S_{51}) r_{11} ::= em & r_{11}
(S_{52}) r_{11} ::= dr & r_{12}
(S_{53}) r_{11} ::= ε

14. Solve: $(F_{14})\ r_{12} = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{dr}\ \&\ r_{12}))))$

$r_{12} = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lid}(\mathrm{dr}\ \&\ r_{12}))))$

$r_{12} = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{dc}\ \&\ \mathrm{lid}(r_{12}))))$ (* Rule I.2.13. *)

We are in the same situation as in **12.**, we get:

$(S_{54})\ r_{12} ::= \mathrm{em}\ \&\ r_{11}$

$(S_{55})\ r_{12} ::= \mathrm{dr}\ \&\ r_{12}$

$(S_{56})\ r_{12} ::= \varepsilon$

15. Solve: $(F_{15})\ r_{13} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],$
$\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ \mathrm{em}\ \&\ r_{13}))))$

$r_{13} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ \mathrm{em}\ \&\ r_{13}))))$

$r_{13} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{ak}(v)\ \&\ \mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))))$
(* Rule I.2.10. *)

Case 15.1.: $\mathrm{car}(\mathrm{med}(\mathrm{ak}(v)\ \&\ \mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))) = \mathrm{em}$

$r_{13} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{ak}(v)\ \&\ \mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))))$

$r_{13} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{em}\ \&\ \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))))$ (* Rule I.3.2. *)

$r_{13} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s],\ \mathrm{non}(v))\ \&\ \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))))$

Case 15.1.1.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]),\ \mathrm{non}(v))\ \&\ \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],$
$\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13})))))$
$= \mathrm{em}$

$r_{13} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s],\ \mathrm{non}(v))\ \&\ \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))))$

$r_{13} = \mathrm{em}\ \&\ \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13}))))$ (* Rule I.3.2. *)

$r_x = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],\ \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ \mathrm{em}\ \&\ r_x))))$

The recursive equation is the same as (F_{15}).

$(S_{57})\ r_{13} ::= \mathrm{em}\ \&\ r_{13}$

Case 15.1.2.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]),\ \mathrm{non}(v))\ \&\ \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],$
$\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)],\ r_{13})))))$
$= \mathrm{dt}(\mathrm{car}(\sigma[s]),\ \mathrm{non}(v))$

$r_{13} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_{13}))))$
$r_{13} = \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_{13}))))$
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} r_x))))$

The recursive equation is the same as (F_{16}).with its solution r_{14}

(S_{58}) r_{13} ::= $\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} r_{14}$

Case 15.2.: $\mathrm{car}(\mathrm{med}(\mathrm{ak}(v) \mathbin{\&} \mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_{13}))) = \mathrm{ak}(v)$

$r_{13} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{ak}(v) \mathbin{\&} \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_{13}))))$
$r_{13} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], r_{13}))))$
(* Rule I.1.9. *)

We are in the same situation as in **14.2.**

16. Solve: (F_{16}) $r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)],$
$\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} r_{14}))))$
$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow \mathrm{non}(v)], \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} r_{14}))))$
$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{ak}(\mathrm{non}(v)) \mathbin{\&} \mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.2.8. *)

Case 16.1.: $\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{non}(v)) \mathbin{\&} \mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))) = \mathrm{em}$

$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{ak}(\mathrm{non}(v)) \mathbin{\&} \mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{em} \mathbin{\&} \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.3.2. *)
$r_{14} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
(* Rule I.1.7. *)

Case 16.1.1.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v))$
$\mathbin{\&} \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))) = \mathrm{em}$

$r_{14} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \mathbin{\&} \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{em} \mathbin{\&} \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.3.2. *)
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], \mathrm{em} \mathbin{\&} r_x))))$

The recursive equation is the same as (F_9).with its solution r_7

(S_{59}) r_{14} ::= $\mathrm{em} \mathbin{\&} r_7$

Case 16.1.2.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v))$
$\quad\& \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14})))))$
$= \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v))$

$r_{14} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \& \mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \& \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
(* Rule I.3.1. *)
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \& r_x))))$

The recursive equation is the same as (F_{10}).with its solution r_8

(S_{60}) $r_{14} ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{non}(v)) \& r_8$

Case 16.2.: $\mathrm{car}(\mathrm{med}(\mathrm{ak}(\mathrm{non}(v)) \& \mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))) = \mathrm{ak}(\mathrm{non}(v))$

$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{med}(\mathrm{ak}(\mathrm{non}(v)) \& \mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{ak}(\mathrm{non}(v)) \& \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.3.1. *)

Case 16.2.1.: $\mathrm{cdr}(\sigma[s]) \neq \varepsilon$

$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{ak}(\mathrm{non}(v)) \& \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v)$
$\quad\& \mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$

Case 16.2.1.1.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \&$
$\quad\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14})))))$
$= \mathrm{em}$

$r_{14} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \& \mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{em} \& \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], \mathrm{em} \& r_x))))$

The recursive equation is the same as (F_{15})

(S_{61}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow r_{14}(\sigma[b \leftarrow \mathrm{non}(v)], \sigma'[b \leftarrow \mathrm{non}(v)])$
$\quad ::= \mathrm{em} \&\ r_{13}(\sigma[b \leftarrow v], \sigma'[b \leftarrow v])$

Case 16.2.1.2.: $\mathrm{car}(\mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \&$

$$\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14})))))$$
$$= \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v)$$

$r_{14} = \mathrm{med}(\mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \;\&\; \mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \;\&\; \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_x = \mathrm{med}(\mathrm{itt}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), v)], \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \;\&\; r_x))))$

(S_{62}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow r_{14}(\sigma[b \leftarrow \mathrm{non}(v)], \sigma'[b \leftarrow \mathrm{non}(v)])$
$::= \mathrm{dt}(\mathrm{car}(\mathrm{cdr}(\sigma[s])), v) \;\&\; r_{14}(\sigma[b \leftarrow v], \sigma'[b \leftarrow v])$

Case 16.2.2.: $\mathrm{cdr}(\sigma[s]) = \varepsilon$

$r_{14} = \mathrm{med}(\mathrm{itt}(\sigma[b \leftarrow \mathrm{non}(v)], \mathrm{ak}(\mathrm{non}(v)) \;\&\; \mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{med}(\mathrm{dr} \;\&\; \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.1.6. *)

Case 16.2.2.1.: $\mathrm{car}(\mathrm{med}(\mathrm{dr} \;\&\; \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))) = \mathrm{em}$

$r_{14} = \mathrm{med}(\mathrm{dr} \;\&\; \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{em} \;\&\; \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.3.2. *)
$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], \mathrm{em} \;\&\; r_x))))$

The recursive equation is the same as (F_{11}) with its solution r_9
(S_{63}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_{14} ::= \mathrm{em} \;\&\; r_9$

Case 16.2.2.2.: $\mathrm{car}(\mathrm{med}(\mathrm{dr} \;\&\; \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))) = \mathrm{dr}$

$r_{14} = \mathrm{med}(\mathrm{dr} \;\&\; \mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$
$r_{14} = \mathrm{dr} \;\&\; \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], r_{14}))))$ (* Rule I.3.1. *)
$r_x = \mathrm{med}(\mathrm{itd}(\mathrm{med}(\mathrm{lit}(\sigma'[b \leftarrow v], \mathrm{dr} \;\&\; r_x))))$

The recursive equation is the same as (F_{12}) with its solution r_{10}
(S_{64}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_{14} ::= \mathrm{dr} \;\&\; r_{10}$

We list the results of our computation. We got 16 recursive equations and 64 relations between their solutions.

(F_1) $c_5 = \mathrm{med}(\mathrm{con}(\sigma, \mathrm{med}(\mathrm{lic}(c_5))))$
(F_2) $r_0 = \mathrm{med}(\mathrm{itc}(\sigma, \mathrm{med}(\mathrm{lic}(\sigma', \mathrm{em} \;\&\; r_0))))$

$(F_3)\ r_1 = med(itc(\sigma, med(lic(\sigma', cr \,\&\, r_1))))$

$(F_4)\ r_2 = med(itc(\sigma, med(lib(em \,\&\, r_2))))$

$(F_5)\ r_3 = med(itc(\sigma, med(lib(\sigma', cr \,\&\, r_3))))$

$(F_6)\ r_4 = med(itt(\sigma[b \leftarrow tt], med(lib(\sigma', dt(car(\sigma[s]), tt) \,\&\, r_4))))$

$(F_7)\ r_5 = med(itd(med(lib(\sigma', em \,\&\, r_5))))$

$(F_8)\ r_6 = med(itd(med(lib(\sigma', dr \,\&\, r_6))))$

$(F_9)\ r_7 = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], em \,\&\, r_7))))$

$(F_{10})\ r_8 = med(itt(\sigma[b \leftarrow v], med(lit(\sigma'[b \leftarrow non(v)], dt(car(\sigma[s]), v) \,\&\, r_8))))$

$(F_{11})\ r_9 = med(itd(med(lit(\sigma'[b \leftarrow non(v)], em \,\&\, r_9))))$

$(F_{12})\ r_{10} = med(itd(med(lit(\sigma'[b \leftarrow non(v)], dr \,\&\, r_{10}))))$

$(F_{13})\ r_{11} = med(itd(med(lid(em \,\&\, r_{11}))))$

$(F_{14})\ r_{12} = med(itd(med(lid(dr \,\&\, r_{12}))))$

$(F_{15})\ r_{13} = med(itt(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), non(v))], med(lit(\sigma'[b \leftarrow non(v)], em \,\&\, r_{13}))))$

$(F_{16})\ r_{14} = med(itt(\sigma[(s, b) \leftarrow (cdr(\sigma[s]), non(v))], med(lit(\sigma'[b \leftarrow non(v)],$
$\qquad\qquad dt(car(cdr(\sigma[s])), non(v)) \,\&\, r_{14}))))$

$(S_1)\ c_5 ::= em \,\&\, r_0$

$(S_2)\ c_5 ::= cr \,\&\, r_1$

$(S_3)\ r_0 ::= em \,\&\, r_0$

$(S_4)\ r_0 ::= cr \,\&\, r_1$

$(S_5)\ r_1 ::= em \,\&\, r_2$

$(S_6)\ r_1 ::= cr \,\&\, r_3$

$(S_7)\ \sigma[s] \neq \varepsilon \Rightarrow r_1 ::= em \,\&\, r_2$

$(S_8)\ \sigma[s] \neq \varepsilon \Rightarrow r_1(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, \&\, r_3(\sigma[b \leftarrow tt], \sigma')$

$(S_9)\ \sigma[s] = \varepsilon \Rightarrow r_1 ::= em \,\&\, r_5$

$(S_{10})\ \sigma[s] = \varepsilon \Rightarrow r_1 ::= dr \,\&\, r_6$

$(S_{11})\ r_2 ::= em \,\&\, r_2$

$(S_{12})\ r_2 ::= cr \,\&\, r_3$

$(S_{13})\ \sigma[s] \neq \varepsilon \Rightarrow r_2 ::= em \,\&\, r_2$

$(S_{14})\ \sigma[s] \neq \varepsilon \Rightarrow r_2(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, \&\, r_3(\sigma[b \leftarrow tt], \sigma')$

$(S_{15})\ \sigma[s] = \varepsilon \Rightarrow r_2 ::= em \,\&\, r_5$

$(S_{16})\ \sigma[s] = \varepsilon \Rightarrow r_2 ::= dr \,\&\, r_6$

$(S_{17})\ r_3 ::= em \,\&\, r_2$

$(S_{18})\ r_3 ::= cr \,\&\, r_3$

$(S_{19})\ \sigma[s] \neq \varepsilon \Rightarrow r_3 ::= em \,\&\, r_2$

$(S_{20})\ \sigma[s] \neq \varepsilon \Rightarrow r_3(\sigma, \sigma') ::= dt(car(\sigma[s]), tt) \,\&\, \&\, r_3(\sigma[b \leftarrow tt], \sigma')$

$(S_{21})\ \sigma[s] = \varepsilon \Rightarrow r_3 ::= em \,\&\, r_5$

(S_{22}) $\sigma[s] = \varepsilon \Rightarrow r_3 ::= \mathrm{dr} \;\&\; r_6$

(S_{23}) $r_4(\sigma[b \leftarrow \mathrm{tt}], \sigma') ::= \mathrm{em} \;\&\; r_7(\sigma[b \leftarrow \mathrm{tt}], \sigma'[b \leftarrow \mathrm{ff}])$

(S_{24}) $r_4(\sigma[b \leftarrow \mathrm{tt}], \sigma') ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), \mathrm{tt}) \;\&\; r_8(\sigma[b \leftarrow \mathrm{tt}], \sigma'[b \leftarrow \mathrm{ff}])$

(S_{25}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= \mathrm{em} \;\&\; r_7(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$

(S_{26}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), v) \;\&\; r_8(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$

(S_{27}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= \mathrm{em} \;\&\; r_9(\sigma'[b \leftarrow \mathrm{ff}])$

(S_{28}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_4(\sigma[b \leftarrow v], \sigma') ::= \mathrm{dr} \;\&\; r_{10}(\sigma'[b \leftarrow \mathrm{ff}])$

(S_{29}) $r_5 ::= \mathrm{em} \;\&\; r_5$

(S_{30}) $r_5 ::= \mathrm{dr} \;\&\; r_6$

(S_{31}) $r_6 ::= \mathrm{em} \;\&\; r_{11}$

(S_{32}) $r_6 ::= \mathrm{dr} \;\&\; r_{12}$

(S_{33}) $r_6 ::= \varepsilon$

(S_{34}) $r_7 ::= \mathrm{em} \;\&\; r_7$

(S_{35}) $r_7 ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), v) \;\&\; r_8$

(S_{36}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow$

$r_7(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$

$::= \mathrm{em} \;\&\; r_{13}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \sigma'[b \leftarrow \mathrm{non}(v)])$

(S_{37}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow$

$r_7(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$

$::= \mathrm{em} \;\&\; r_{14}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \sigma'[b \leftarrow \mathrm{non}(v)])$

(S_{38}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_7 ::= \mathrm{em} \;\&\; r_9$

(S_{39}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_7 ::= \mathrm{dr} \;\&\; r_{10}$

(S_{40}) $r_8 ::= \mathrm{em} \;\&\; r_7$

(S_{41}) $r_8 ::= \mathrm{dt}(\mathrm{car}(\sigma[s]), v) \;\&\; r_8$

(S_{42}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow$

$r_8(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$

$::= \mathrm{em} \;\&\; r_{13}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \sigma'[b \leftarrow \mathrm{non}(v)])$

(S_{43}) $\mathrm{cdr}(\sigma[s]) \neq \varepsilon \Rightarrow$

$r_8(\sigma[b \leftarrow v], \sigma'[b \leftarrow \mathrm{non}(v)])$

$::= \mathrm{em} \;\&\; r_{14}(\sigma[(s, b) \leftarrow (\mathrm{cdr}(\sigma[s]), \mathrm{non}(v))], \sigma'[b \leftarrow \mathrm{non}(v)])$

(S_{44}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_8 ::= \mathrm{em} \;\&\; r_9$

(S_{45}) $\mathrm{cdr}(\sigma[s]) = \varepsilon \Rightarrow r_8 ::= \mathrm{dr} \;\&\; r_{10}$

(S_{46}) $r_9 ::= \mathrm{em} \;\&\; r_9$

(S_{47}) $r_9 ::= \mathrm{em} \;\&\; r_{10}$

(S_{48}) $r_{10} ::= \mathrm{em} \;\&\; r_{11}$

(S_{49}) $r_{10} ::= \mathrm{dr} \;\&\; r_{12}$

(S_{50}) r_{10} ::= ε

(S_{51}) r_{11} ::= em & r_{11}

(S_{52}) r_{11} ::= dr & r_{12}

(S_{53}) r_{11} ::= ε

(S_{54}) r_{12} ::= em & r_{11}

(S_{55}) r_{12} ::= dr & r_{12}

(S_{56}) r_{12} ::= ε

(S_{57}) r_{13} ::= em & r_{13}

(S_{58}) r_{13} ::= dt(car(σ[s]), non(v)) & r_{14}

(S_{59}) r_{14} ::= em & r_7

(S_{60}) r_{14} ::= dt(car(σ[s]), non(v)) & r_8

(S_{61}) cdr(σ[s]) $\neq \varepsilon \Rightarrow$ r_{14}(σ[b $\leftarrow$ non(v)], σ'[b $\leftarrow$ non(v)]) ::= em & r_{13}(σ[b $\leftarrow$ v], σ'[b $\leftarrow$ v])

(S_{62}) cdr(σ[s]) $\neq \varepsilon \Rightarrow$ r_{14}(σ[b $\leftarrow$ non(v)], σ'[b $\leftarrow$ non(v)])
::= dt(car(cdr(σ[s])), v) & r_{14}(σ[b $\leftarrow$ v], σ'[b $\leftarrow$ v])

(S_{63}) cdr(σ[s]) = $\varepsilon \Rightarrow$ r_{14} ::= em & r_9

(S_{64}) cdr(σ[s]) = $\varepsilon \Rightarrow$ r_{14} ::= dr & r_{10}

To continue we have to reason about liveness. In this example this is more technical than in the example of App II.
It is not enough to define liveness with the terms:
A system is *live,* if it eventually leaves a state after entering it.
We have to prove: Every state is eventually reached.