We already proved that

$$\text{If } \mathit{nullable}(r) \text{ then } \mathit{POSIX}\ (\mathit{mkeps}\ r)\ r$$

holds. This is essentially the "base case" for the correctness proof of the algorithm. For the "induction case" we need the following main theorem, which we are currently after:

$$\begin{array}{lll} \text{If} & (*) & \mathit{POSIX}\ v\ (\mathit{der}\ c\ r) \text{ and } \vdash v : \mathit{der}\ c\ r \\ \text{then} & & \mathit{POSIX}\ (\mathit{inj}\ r\ c\ v)\ r \end{array}$$

That means a POSIX value v is still POSIX after injection. I am not sure whether this theorem is actually true in this full generality. Maybe it requires some restrictions.

If we unfold the POSIX definition in the then-part, we arrive at

$$\forall v'.\ \text{if } \vdash v' : r \text{ and } |\mathit{inj}\ r\ c\ v| = |v'| \text{ then } |\mathit{inj}\ r\ c\ v| \succ_r v'$$

which is what we need to prove assuming the if-part (*) in the theorem above. Since this is a universally quantified formula, we just need to fix a v'. We can then prove the implication by assuming

$$\text{(a)} \ \ \vdash v' : r \ \ \text{and} \ \ \text{(b)} \ \ \mathit{inj}\ r\ c\ v = |v'|$$

and our goal is

$$(\mathit{goal}) \ \ \mathit{inj}\ r\ c\ v \succ_r v'$$

There are already two lemmas proved that can transform the assumptions (a) and (b) into

$$\text{(a*)} \ \ \vdash \mathit{proj}\ r\ c\ v' : \mathit{der}\ c\ r \ \ \text{and} \ \ \text{(b*)} \ \ c\,\#\,|v| = |v'|$$

Another lemma shows that

$$|v'| = c\,\#\,|\mathit{proj}\ r\ c\ v|$$

Using (b*) we can therefore infer

$$\text{(b**)} \ \ |v| = |\mathit{proj}\ r\ c\ v|$$

The main idea of the proof is now a simple instantiation of the assumption $\mathit{POSIX}\ v\ (\mathit{der}\ c\ r)$. If we unfold the POSIX definition, we get

$$\forall v'.\ \text{if } \vdash v' : der\ c\ r \text{ and } |v| = |v'| \text{ then } v \succ_{der\ c\ r}\ v'$$

We can instantiate this v' with $proj\ r\ c\ v'$ and can use (a*) and (b**) in order to infer

$$v \succ_{der\ c\ r}\ proj\ r\ c\ v'$$

The point of the side-lemma below is that we can "add" an inj to both sides to obtain

$$inj\ r\ c\ v \succ_r\ inj\ r\ c\ (proj\ r\ c\ v')$$

Finally there is already a lemma proved that shows that an injection and projection is the identity, meaning

$$inj\ r\ c\ (proj\ r\ c\ v') = v'$$

With this we have shown our goal (pending a proof of the side-lemma next).

Side-Lemma

A side-lemma needed for the theorem above which might be true, but can also be false, is as follows:

If	(1)	$v_1 \succ_{der\ c\ r} v_2$,
	(2)	$\vdash v_1 : der\ c\ r$, and
	(3)	$\vdash v_2 : der\ c\ r$ holds,
then		$inj\ r\ c\ v_1 \succ_r inj\ r\ c\ v_2$ also holds.

It essentially states that if one value v_1 is bigger than v_2 then this ordering is preserved under injections. This is proved by induction (on the definition of der... this is very similar to an induction on r).

The case that is still unproved is the sequence case where we assume $r = r_1 \cdot r_2$ and also r_1 being nullable. The derivative $der\ c\ r$ is then

$$der\ c\ r = ((der\ c\ r_1) \cdot r_2) + (der\ c\ r_2)$$

or without the parentheses

$$der\ c\ r = (der\ c\ r_1) \cdot r_2 + der\ c\ r_2$$

In this case the assumptions are

(a)	$v_1 \succ_{(der\ c\ r_1)\cdot r_2 + der\ c\ r_2} v_2$
(b)	$\vdash v_1 : (der\ c\ r_1) \cdot r_2 + der\ c\ r_2$
(c)	$\vdash v_2 : (der\ c\ r_1) \cdot r_2 + der\ c\ r_2$
(d)	$nullable(r_1)$

The induction hypotheses are

$$\text{(IH1)} \quad \forall v_1 v_2.\ v_1 \succ_{der\ c\ r_1} v_2 \wedge \vdash v_1 : der\ c\ r_1 \wedge \vdash v_2 : der\ c\ r_1 \longrightarrow inj\ r_1\ c\ v_1 \succ r_1\ inj\ r_1\ c\ v_2$$

$$\text{(IH2)} \quad \forall v_1 v_2.\ v_1 \succ_{der\ c\ r_2} v_2 \wedge \vdash v_2 : der\ c\ r_2 \wedge \vdash v_2 : der\ c\ r_2 \longrightarrow inj\ r_2\ c\ v_1 \succ r_2\ inj\ r_2\ c\ v_2$$

The goal is

$$(goal) \qquad inj\ (r_1 \cdot r_2)\ c\ v_1 \succ_{r_1 \cdot r_2} inj\ (r_1 \cdot r_2)\ c\ v_2$$

If we analyse how (a) could have arisen (that is make a case distinction), then we will find four cases:

LL	$v_1 = Left(w_1),\ v_2 = Left(w_2)$
LR	$v_1 = Left(w_1),\ v_2 = Right(w_2)$
RL	$v_1 = Right(w_1),\ v_2 = Left(w_2)$
RR	$v_1 = Right(w_1),\ v_2 = Right(w_2)$

We have to establish our goal in all four cases.

Case LR

The corresponding rule (instantiated) is:

$$\frac{len\,|w_1| \geq len\,|w_2|}{Left(w_1) \succ_{(der\ c\ r_1)\cdot r_2 + der\ c\ r_2} Right(w_2)}$$

This means we can also assume in this case

$$(e) \quad len\,|w_1| \geq len\,|w_2|$$

which is the premise of the rule above. Instantiating v_1 and v_2 in the assumptions (b) and (c) gives us

(b*)	$\vdash Left(w_1) : (der\ c\ r_1) \cdot r_2 + der\ c\ r_2$
(c*)	$\vdash Right(w_2) : (der\ c\ r_1) \cdot r_2 + der\ c\ r_2$

Since these are assumptions, we can further analyse how they could have arisen according to the rules of $\vdash \_ : \_$. This gives us two new assumptions

$$\begin{array}{ll} (\text{b}^{**}) & \vdash w_1 : (der\ c\ r_1) \cdot r_2 \\ (\text{c}^{**}) & \vdash w_2 : der\ c\ r_2 \end{array}$$

Looking at (b**) we can further analyse how this judgement could have arisen. This tells us that w_1 must have been a sequence, say $u_1 \cdot u_2$, with

$$\begin{array}{ll} (\text{b}^{***}) & \vdash u_1 : der\ c\ r_1 \\ & \vdash u_2 : r_2 \end{array}$$

Instantiating the goal means we need to prove

$$inj\ (r_1 \cdot r_2)\ c\ (Left(u_1 \cdot u_2)) \succ_{r_1 \cdot r_2} inj\ (r_1 \cdot r_2)\ c\ (Right(w_2))$$

We can simplify this according to the rules of *inj*:

$$(inj\ r_1\ c\ u_1) \cdot u_2 \succ_{r_1 \cdot r_2} (mkeps\ r_1) \cdot (inj\ r_2\ c\ w_2)$$

This is what we need to prove. There are only two rules that can be used to prove this judgement:

$$\frac{v_1 = v_1' \qquad v_2 \succ_{r_2} v_2'}{v_1 \cdot v_2 \succ_{r_1 \cdot r_2} v_1' \cdot v_2'} \qquad \frac{v_1 \succ_{r_1} v_1'}{v_1 \cdot v_2 \succ_{r_1 \cdot r_2} v_1' \cdot v_2'}$$

Using the left rule would mean we need to show that

$$inj\ r_1\ c\ u_1 = mkeps\ r_1$$

but this can never be the case.[1] Lets assume it would be true, then also if we flat each side, it must hold that

$$|inj\ r_1\ c\ u_1| = |mkeps\ r_1|$$

But this leads to a contradiction, because the right-hand side will be equal to the empty list, or empty string. This is because we assumed $nullable(r_1)$ and there is a lemma called `mkeps_flat` which shows this. On the other side we know by assumption (b***) and lemma `v4` that the other side needs to be a string starting with c (since we inject c into u_1). The empty string can never be equal to something starting with c... therefore there is a contradiction.

[1] Actually Isabelle found this out after analysing its argument. ;o)

That means we can only use the rule on the right-hand side to prove our goal. This implies we need to prove

$$inj\ r_1\ c\ u_1 \succ_{r_1} mkeps\ r_1$$

Case RL

The corresponding rule (instantiated) is:

$$\frac{len\,|w_1| > len\,|w_2|}{Right(w_1) \succ_{(der\ c\ r_1)\cdot r_2 + der\ c\ r_2} Left(w_2)}$$

Test Proof

We want to prove that

$$nullable(r) \text{ implies } POSIX(mkeps\ r)\ r$$

We prove this by induction on r. There are 5 subcases, and only the $r_1 + r_2$-case is interesting. In this case we know the induction hypotheses are

$$\begin{array}{ll} \text{(IMP1)} & nullable(r_1) \text{ implies } POSIX(mkeps\ r_1)\ r_1 \\ \text{(IMP2)} & nullable(r_2) \text{ implies } POSIX(mkeps\ r_2)\ r_2 \end{array}$$

and know that $nullable(r_1 + r_2)$ holds. From this we know that either $nullable(r_1)$ holds or $nullable(r_2)$. Let us consider the first case where we know $nullable(r_1)$.

Problems in the paper proof

I cannot verify…

Isabelle Cheat-Sheet

- The main notion in Isabelle is a *theorem.* Definitions, inductive predicates and recursive functions all have underlying theorems. If a definition is called `foo`, then the theorem will be called `foo_def`. Take a recursive function, say `bar`, it will have a theorem that is called `bar.simps` and will be added to the simplifier. That means the simplifier will automatically Inductive predicates called `baz` will be called `baz.intros`. For inductive predicates, there are also theorems `baz.induct` and `baz.cases`.

- A *goal-state* consists of one or more subgoals. If there are `No more subgoals!` then the theorem is proved. Each subgoal is of the form

$$[\![\ldots premises \ldots]\!] \Longrightarrow conclusion$$

  where *premises* and *conclusion* are formulas of type `bool`.

- There are three low-level methods for applying one or more theorem to a subgoal, called `rule`, `drule` and `erule`. The first applies a theorem to a conclusion of a goal. For example

$$\texttt{apply}(\texttt{rule}\ thm)$$

  If the conclusion is of the form $\_ \wedge \_$, $\_ \longrightarrow \_$ and $\forall\, x.\_$ the *thm* is called

$$\begin{array}{lcl} \_ \wedge \_ & \Rightarrow & conjI \\ \_ \longrightarrow \_ & \Rightarrow & impI \\ \forall\, x.\_ & \Rightarrow & allI \end{array}$$

  Many of such rule are called intro-rules and end with an "*I*", or in case of inductive predicates *_.intros*.