# Formal Methods: Lecture 6

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

## Checking Confluence: the Church-Rosser Checker

The Maude formal tool environment includes the Church-Roser Checker, implemented by Francisco Durán, which, under the assumption of termination, tries to check the confluence property.

The tool may succeed, in which case it responds with a confirmation that the module is confluent, or it may fail to check it, in which case it respons with a set of proof obligations required to ensure that the specification is ground confluent.

The most recent version of the tool can handle Maude functional modules with any combination of `assoc`, `comm`, and `id:` attributes, but may fail to handle some `assoc` without `comm` operators.

## Checking Confluence: the Church-Rosser Checker (II)

If we submit to the tool a module `fmod`$(\Sigma, E)$`endfm`, where the only possible equational attribute in operators is `comm`, and such that the equations $E$ are unconditional, then if the equations $E$ are indeed sort-decreasing and confluent, the checker tool will succeed.

Failure to pass the check may mean one of three things:

- $(\Sigma, E)$ is ground confluent, but not confluent;
- $(\Sigma, E)$ is confluent, but further reasoning is needed, because some of the equations in $E$ are conditional;
- $(\Sigma, E)$ fails to be ground confluent.

## Checking Confluence: the Church-Rosser Checker (III)

In case the check fails, the proof obligations returned can be very useful for further analysis, either to establish the property, or to find a conterexample.

The Curch-Rosser Checker is a Maude program available on the Maude web page. It extends Full Maude, and checks the confluence of Full Maude functional modules (assuming termination).

In Full Maude we should always enter modules enclosed in parentheses. After entering the module, we then give to the Church-Rosser Checker the command,

```
Maude> (check Church-Rosser .)
```

also enclosed in parentheses.

## Checking Confluence: the Church-Rosser Checker (IV)

We can illustrate the use of the Church-Rosser Checker with our running example of natural number addition enclosed in parentheses.

```
(fmod NAT-MIXFIX is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s_ : Natural -> Natural [ctor] .
  op _+_ : Natural Natural -> Natural .
  vars N M : Natural .
  eq N + 0 = N .
  eq N + s M = s(N + M) .
  endfm)
```

## Checking Confluence: the Church-Rosser Checker (V)

```
Maude> in nat-mixfix

Introduced module: NAT-MIXFIX
Maude> (check Church-Rosser .)

Church-Rosser checking of NAT-MIXFIX
Checking solution :
The specification is Church-Rosser .
```

## Ground Confluent but not Confluent

Quite often, a module will not pass the check, not because there is any real problem with its equations, but simply because it is ground confluent but not confluent.

In such a case, the tool will return a set of critical pairs as proof obligations. Such critical pairs are equations $(\forall X)\, t = t'$ such that:

- $E \vdash (\forall X)\, t = t'$,
- the tool failed to establish $t \downarrow_E t'$.

Furthemore, as we shall see, they are sufficient, as proof obligations, to establish ground confluence. That is, if we can show $\overline{\theta}(t) \downarrow_E \overline{\theta}(t')$ for each ground substitution $\theta$, then $E$ is indeed ground confluent (assuming termination).

## Ground Confluent but not Confluent (II)

We can illustrate ground confluence with the following module,

```
(fmod ANOTHER-NAT is
   sorts Zero Natural .
   subsort Zero < Natural .
   op 0 : -> Zero .
   op s_ : Natural -> Natural .
   ops (_+_) (_*_) : Natural Natural -> Natural [comm] .
   vars N M : Natural .
   eq 0 + N = N .
   eq s N + M = s (N + M) .
   eq 0 * N = 0 .
   eq s N * M = M + (N * M) .
 endfm)
```

## Ground Confluent but not Confluent (II)

```
Maude> in ex-crc-nats.fm
rewrites: 1293 in 20ms cpu (30ms real) (64650 rewrites/second)

Introduced module: ANOTHER-NAT
rewrites: 5452 in 20ms cpu (30ms real) (272600 rewrites/second)

Church-Rosser checking of  ANOTHER-NAT
Checking solution :
var N : Natural .
var N@ : Natural .
cp s ( N + ( N@ + ( N * N@ ) ) ) = s ( N@ + ( N + ( N * N@ ) ) )
rewrites: 1368 in 0ms cpu (10ms real) (~ rewrites/second)
```

## Ground Confluent but not Confluent (II)

Where does this critical pair come from? It comes from applying the equation

```
eq s N * M = M + (N * M) .
```

modulo commutativity to the term `s N * s M` in two different ways yielding terms that, after further simplification,

```
s M + (N * s M) = s(M + (N + (N * M)))
```

```
s N + (M * s N) = s(N + (M + (N * M)))
```

cannot be further simplified, and therefore cannot be joined, showing that the equations are not confluent. However, every ground instance can be joined.

## Ground Confluent but not Confluent (III)

What can we do in such a situation? One of four things:

1. use the critical pair as useful information to transform the equations into equivalent equations that are confluent and pass the test; or

2. attempt an automatic transformation into confluent equations using Maude's Knuth-Bendix Completion tool; or

3. prove an inductive theorem about the rewriting relation $\longrightarrow_E$ itself, not about equality!, showing that for each ground instance the pair can be joined; or

4. find a conterexample disproving ground confluence.

## Ground Confluent but not Confluent (IV)

In our example, alternative (1) yields a transformed module, by realizing that the equation

```
eq s N * M = M + (N * M) .
```

is in a sense too general, since, the case `M = 0` is covered by the other equations for addition and multiplication. Therefore, we can assume `M = s M'`, and replace the above equation by the more specialized equation,

```
eq s N * s M = s((N + M) + (N * M)) .
```

to get the confluent transformed module,

## Ground Confluent but not Confluent (V)

```
(fmod CONFLUENT-NAT is
   sorts Zero Natural .
   subsort Zero < Natural .
   op 0 : -> Zero .
   op s_ : Natural -> Natural .
   ops (_+_) (_*_) : Natural Natural -> Natural [comm] .
   vars N M : Natural .
   eq 0 + N = N .
   eq s N + M = s (N + M) .
   eq 0 * N = 0 .
   eq s N * s M = s((N + M) + (N * M)) .
 endfm)
```

## Ground Confluent but not Confluent (VI)

```
Maude> in ex-crc-nats-revised.fm

Introduced module: CONFLUENT-NAT
Maude> (check Church-Rosser .)

Church-Rosser checking of CONFLUENT-NAT
Checking solution :
The specification is Church-Rosser .
```

## Justification of the Church-Rosser Checker

The justification will be somewhat incomplete, yet hopefully intuitive. Fuller accounts of the issues involved can be found in:

- F. Durán and J. Meseguer, "A Church-Rosser Checker Tool for Maude Specifications," 2000, in `http://maude.csl.sri.com`

- A. Bouhoula, J.-P. Jouannaud, and J. Meseguer, "Secification and Proof in Membership Equational Logic," Theoretical Computer Science, 236 (1): 35–132, 2000.

## Justification of the Church-Rosser Checker (II)

The Church-Rosser Checker does two things:

- check that the equations are sort-decreasing; and
- check confluence (assuming termination) by checking all possible critical pairs.

Checking sort-decreasingness is relatively easy to do, since it only depends on the sorts of the terms we substitute for the variables in any instance, not on the terms themselves. Therefore, we may as well substitute terms that are variables of smaller sorts.

Substitutions of this kind, mapping variables to other variables of smaller sorts, are called variable specializations and (up to renaming) there is only a finite number of such specializations.

## Justification of the Church-Rosser Checker (III)

Consider, for example, that we want to check the sort-decreasingness of an equation, with `I` of sort `Integer`,

```
eq I + 0 = I .
```

in a module `INTEGER` with two declarations of addition,

```
op _+_ : Natural Natural -> Natural .
op _+_ : Integer Integer -> Integer .
```

and with subsorts `NzInteger`, `NzNatural`, and `Natural`. Then, to check that the equation is sort decreasing, it is enough to check that the sort of the lefthand side is greater of equal to that of the righthand side for the original equation, and for the equations obtained replacing `I` by a variable in each of the subsorts.

## Justification of the Church-Rosser Checker (IV)

So, the main questions remaining are:

- what is a critical pair? and
- why is checking joinability of critical pairs sufficient for checking confluence under the termination assumption?

## Justification of the Church-Rosser Checker (V)

The key observation is the subject of **Ex.**6.2, namely, that a set $E$ of equations that is locally confluent and terminating is confluent.

Therefore, under the termination assumption, all we need to do is to convince ourselves that, given a term $t$, and given two one-step simplifications $t \longrightarrow_E t'$, and $t \longrightarrow_E t''$, we always can join $t'$ and $t''$. That is, we always have, $t' \downarrow_E t''$.

The crucial point, then, is to analyze where in $t$ do the rewrites $t \longrightarrow_E t'$, and $t \longrightarrow_E t''$ happen.

## Justification of the Church-Rosser Checker (VI)

Recall that a simplification step with $E$ must happen at a given position $\pi$ in $t$. Therefore, there must be two positions, $\pi$ and $\rho$ in $t$ and two equations (to simplify life let us assume they are unconditional, but all generalizes to the conditional case) $u = v$ and $u' = v'$ in $E$ with substitutions $\theta$ and $\mu$ such that:

- $t = t[\pi \leftarrow \overline{\theta}(u)] = t[\rho \leftarrow \overline{\mu}(u')]$;
- $t' = t[\pi \leftarrow \overline{\theta}(v)]$;
- $t'' = t[\rho \leftarrow \overline{\mu}(v')]$.

all now hinges upon where $\pi$ and $\rho$ are located in $t$.

## Justification of the Church-Rosser Checker (VII)

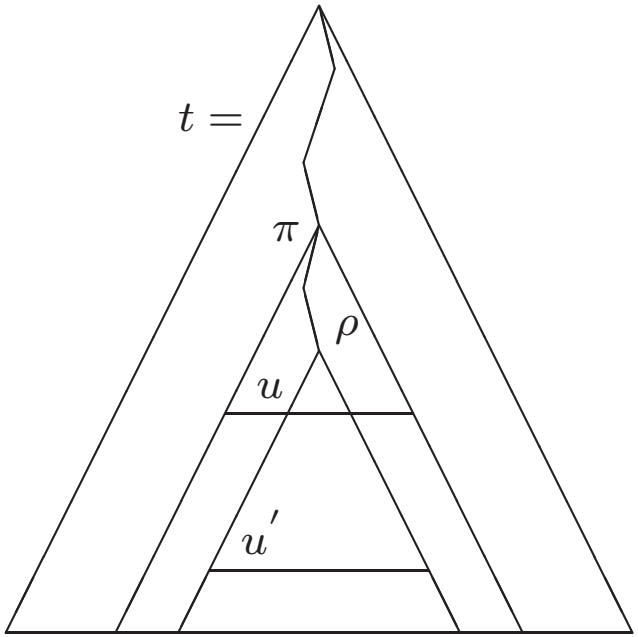
There are essentially three possibilities (see the picture):

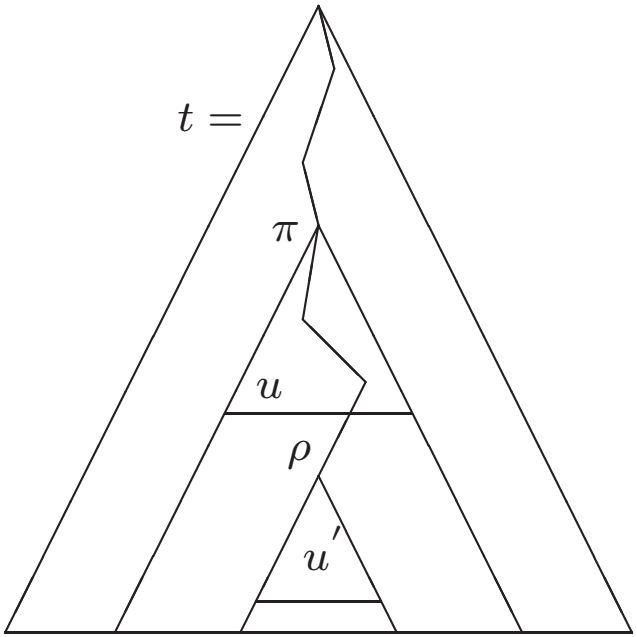
1. **nested simplification with overlap**: there is a path $\nu$ such that $\rho = \pi.\nu$ (or $\pi = \rho.\nu$, but this case is symmetric) and $\nu$ is a nonvariable position in $u$;

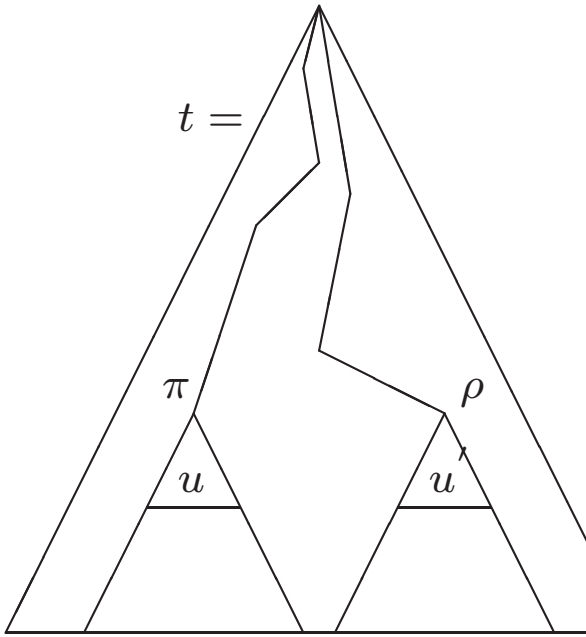
2. **nested simplification without overlap**: there is a path $\nu$ such that $\rho = \pi.\nu$, but $\nu$ is not a nonvariable position in $u$;

3. **sideways simplification**: there isn't such a $\nu$ at all.

## Three Possibilities for $\pi$ and $\rho$



## Justification of the Church-Rosser Checker (VIII)

In the sideways case, where neither $\rho = \pi.\nu$, nor $\pi = \rho.\nu$, the positions are totally independent, in the sense that we have:

$$t = (t[\pi \leftarrow \overline{\theta}(u)])[\rho \leftarrow \overline{\mu}(u')] = (t[\rho \leftarrow \overline{\mu}(u')])[\pi \leftarrow \overline{\theta}(u)].$$

Therefore, we have:

- $t' = t[\pi \leftarrow \overline{\theta}(v)] = (t[\rho \leftarrow \overline{\mu}(u')])[\pi \leftarrow \overline{\theta}(v)] = (t[\pi \leftarrow \overline{\theta}(v)])[\rho \leftarrow \overline{\mu}(u')]$; and

- $t'' = t[\rho \leftarrow \overline{\mu}(v')] = (t[\pi \leftarrow \overline{\theta}(u)])[\rho \leftarrow \overline{\mu}(v')] = (t[\rho \leftarrow \overline{\mu}(v')])[\pi \leftarrow \overline{\theta}(u)]$.

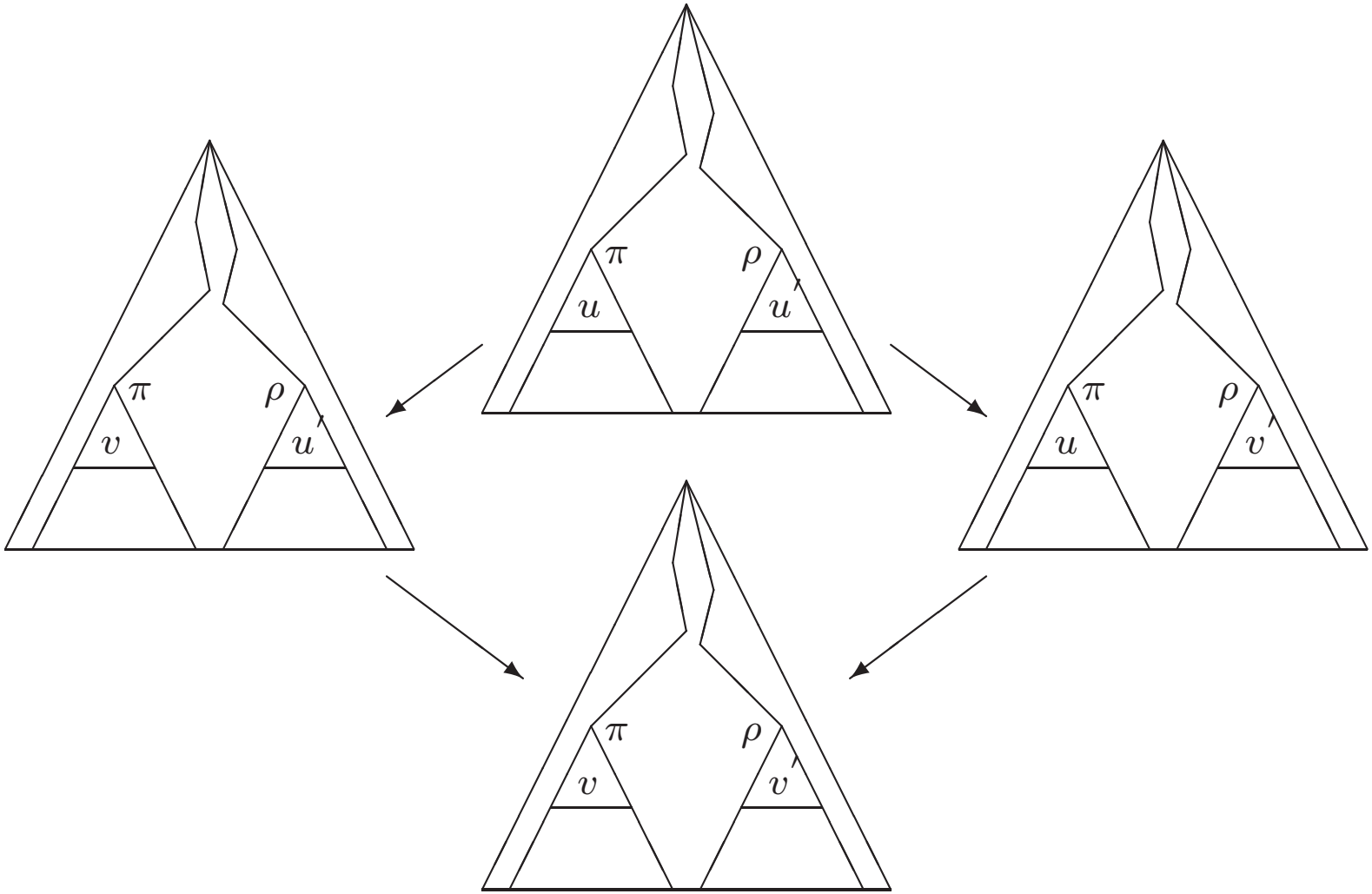
## Justification of the Church-Rosser Checker (IX)

Therefore we have a term $w$ of the form (see the picture):

$$w = t'[\rho \leftarrow \overline{\mu}(v')] = (t[\pi \leftarrow \overline{\theta}(v)])[\rho \leftarrow \overline{\mu}(v')] = t''[\pi \leftarrow \overline{\theta}(v)].$$

and therefore, $t' \longrightarrow_E w$, and $t'' \longrightarrow_E w$.

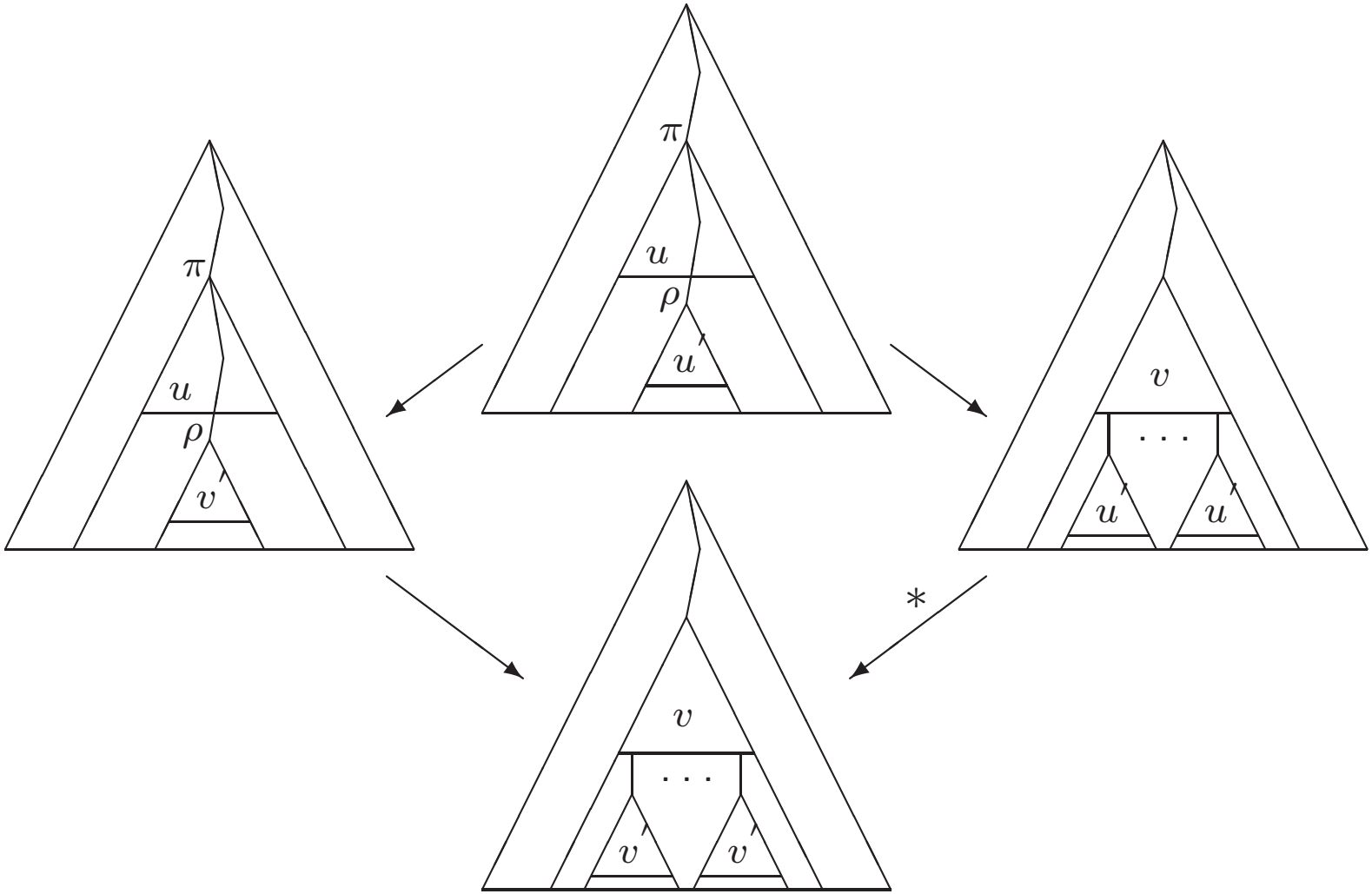
# Sideways Simplification



## Justification of the Church-Rosser Checker (X)

The case of nested simplification without overlap is also always joinable. The detailed proof is left as an exercise; but note that, since a single variable in $u$ may occur several times in $v$, the occurrence of $u'$ underneath $u$ may be copied several times by the simplification with the equation $u = v$. This means that to reach a common $w$ from $t'$ several steps of simplification may be needed (see the picture).

# Nested Simplification



## Justification of the Church-Rosser Checker (XI)

Therefore, we have so far reduced the problem of confluence (under the termination assumption) to the considerably simpler problem of joinability of nested simplifications with overlap, and the question still remains,

What is a critical pair? Stay tuned!

## Readings and Exercises

Read the report,

F. Durán and J. Meseguer, "A Church-Rosser Checker Tool for Maude Specifications," 2000,

which can be found in the Maude web page. Download the Church-Rosser Checker tool from the Maude web page. Then,

- check the confluence of several simple examples of your own, and
- if the tool responds with some critical pairs, analyze such pairs to understand how those critical pairs were originated and attempt a hand transformation of the specification into an equivalent one that is confluent.

## Exercises (II)

**Ex.** 6.1 Call an ordered set $(A, \leq)$ noetherian iff any nonempty subset $B \subseteq A$ has a minimal element, that is, an element $x \in B$ such that $\forall y \in B\ x \geq y \Rightarrow x = y$.

- prove that the following principle of noetherian induction is correct:

  "If $(A, \leq)$ is noetherian, then any $B \subseteq A$ such that for any $a \in A$ if $a\!\downarrow \subseteq B$ then $a \in B$ must be $B = A$, where, by definition, $a\!\downarrow = \{x \in A \mid x < a\}$."

- prove that $(A, \leq)$ is noetherian if and only if there are no infinite strictly descending chains in $(A, \leq)$ of the form

$$a_0 > a_1 > a_2 > \ldots a_n > \ldots$$

## Exercises (III)

- Show that if a set of equations $E$ is terminating, then for any $s \in S$, the relation $\stackrel{*}{\longrightarrow}_E$ defines a noetherian partial order on $T_\Sigma(X)_s$.

**Ex.** 6.2 Call a set $E$ of equations locally confluent iff whenever we have $t \longrightarrow_E u$ and $t \longrightarrow_E v$, then $u \downarrow_E v$.

- prove that if $E$ is terminating and locally confluent, then $E$ is confluent (Hint: use noetherian induction)

- exhibit a set of equations that are localy confluent but not confluent.

## Exercises (IV)

**Ex.** 6.3 Call $E$ weakly terminating if any term $t$ can be rewritten by $E$ to a term $t'$ that cannot be further rewritten.

- show that if $E$ is weakly terminating and confluent, there is a unique canonical form $can_E(t)$ for any $t$

- exhibit a set of equations $E$ that is weakly terminating and confluent, but not terminating.