Extended Summary of the MAC-lecture September 1

September 2, 2003

Map is a computer program that we may use to let the computer read the mathematical (algebraic) system in which we want to operate, i.e. do our mathematics (e.g. proofs).
When the algebraic system has been read by the computer, it is capable of checking the mathematics we do in that system (by means of the Map program).

Mac is the mathematical system used in this course. It is actually a so-called derivation system, but like algebraic systems it consists of the following three types af elements:

a) Constructs (i.e. building blocks for mathematical expressions):

- **operators**, such as numbers, variables, +, -, $\cdot$, = , etc.
  Operators take 0, 1, 2 or more arguments and return a **value**.
  Numbers and variables are nullary operators, 'minus' may be unary (as in -x) or binary (as in x - y).
  The position of the operator may be prefix (-x), infix (x-y), suffix (x^+), "outfix" (2), or none of these ($\frac{x}{y}$).
- **directives**, such as $\equiv$, $\breve{>}$, $\breve{=}$, $\ddot{=}$, etc.
  Directives are used for informing the computer about the algebraic system.
  Map has e.g. received the information that x $\equiv$ x holds for any Mac-expression x ([Mac rule Reflexivity: x $\equiv$ x] on page 580 has been read by Map) and thus Map knows that 1/0 $\equiv$ 1/0 holds in the Mac System.
  The priority list on page 680 and the associativity rules on page 679 have also been read by Map (at least in theory, since it is not yet fully operative). Hence Map knows e.g. that the following Mac expresions hold
  [$x^+ \breve{>} x \cdot y \breve{>} -x \breve{>} x + y \breve{=} x - y$],
  [$x + y + z \breve{\rightarrow} (x + y) + z$] (i.e. + is left associative),
  [$x - y - z \breve{\rightarrow} (x - y) - z$] (since [$x + y \breve{=} x - y$] implies same associativity for + and -) and
  [$x + y - z \breve{\rightarrow} (x + y) - z$] (since [$x + y \breve{=} x - y$]).
- **macros/shorthands for Mac expressions**, such as the parenthesis in (4 + 3).
  Using the directive $\ddot{=}$ Map has received the macro-definition [(x) $\ddot{=}$ x] which it may use when it has built the parse tree *after* the reading (parse) of an expression containing parenthesis.

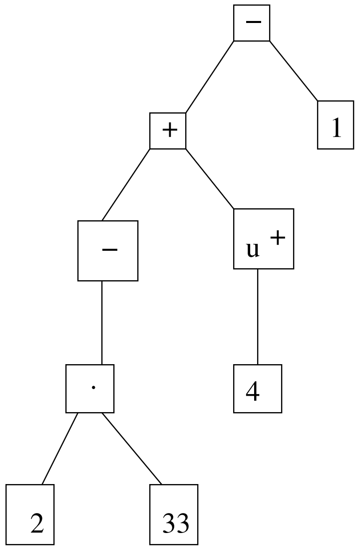
b) One antirule (i.e. a rule stating something that does **not** hold, i.e. fails, in the algebraic system)
Using the directive [Mac antirule x:y]$^{\cdot}$ Map has been informed that [Contradiction]$^{\cdot}$ is a macro for the expression [$\perp\equiv$ T]$^{\cdot}$, and that the latter expression fails
(i.e. $\perp$ and T are **not** equal in the Mac System). Map read the following expression:
[**Mac antirule Contradiction**: $\perp\equiv$ T] (on page 580).

c) Some fundamental rules/axioms , e.g. Mac rule Reflexivity mentioned in **a)** above.

An expression has a value if and only if it consists of operators/macros for expressions with value, and the constructs are placed syntactically correct. Expressions with values are called **terms** whereas the other expressions are called **statements**.

Knowing the priorities of the operators/directives (p.680) and the associativity rules (p.679) Map builds a **parse tree** when reading/parsing an expression.

Example 1. Reading $[-2 \cdot 33 + 4^{+} - 1]^{\cdot}$ Map builds the parse tree

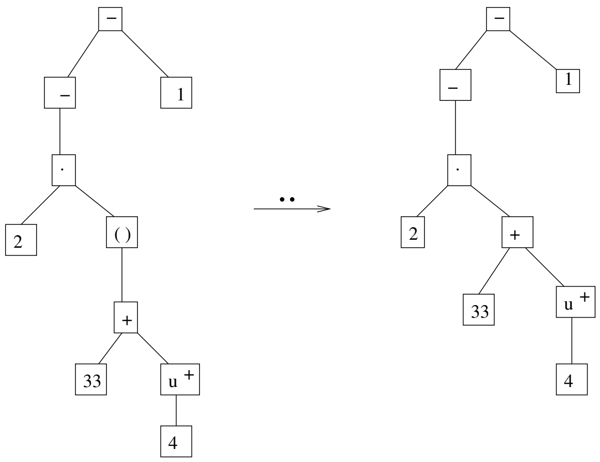


since subtraction of 1 is the last thing to do, i.e. the last minus in the expression is the **principal operator**, i.e. the operator in the **root** of the tree. This tree contains much more information than the original expression! We see e.g. that the numbers are nullary operators, whereas $\cdot$ is binary. We also see e.g. that $x \cdot y$ has higher priority than $-x$. Since all the **nodes** in the tree are operators, it is now easy to compute the value **from below and up to the root**. The operator x^{+} adds 1 to its argument and thus the following equation (Mac expression) holds

$$-2 \cdot 33 + 4^{+} - 1 \equiv ((-(2 \cdot 33)) + (4^{+})) - 1 \equiv -62$$

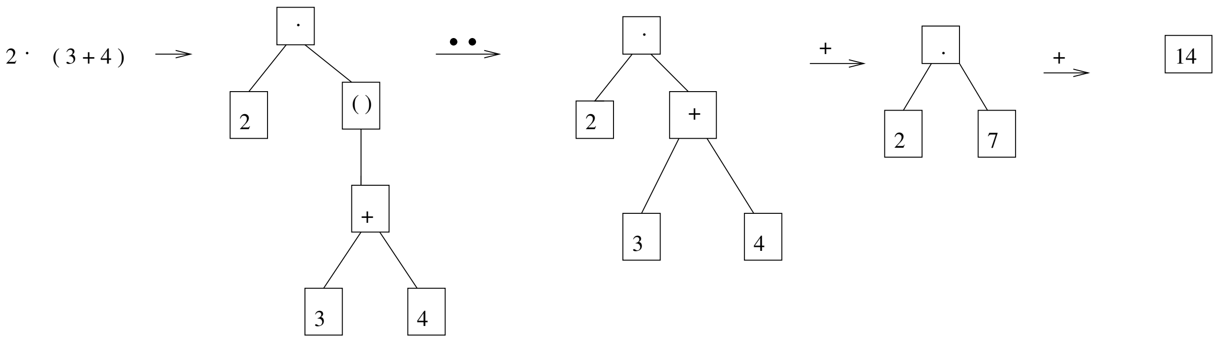
Trees containing merely operators are called **syntax trees**.

Example 2. Reading $[-2 \cdot (33 + 4^{+}) - 1]^{\cdot}$ Map builds a parse tree and then uses the macro definition $[\ (x) \ddot{=} x\ ]$ to obtain a syntax tree (containing operators) from which the value -77 is easily calculated.



Note that the parse tree contains the constructs that Map reads (parses), whereas a further transformation to a syntax tree is only possible if a term has been read. Expressions with *syntactically incorrect* placement of constructs as e.g. $[\cdot + ()\ 2]^{o}$ or just $[2x]^{o}$ does not even have a parse tree (Map complains during the parse!).

Different types of arrows are used to describe **parse reductions** ($\rightarrow$), **macro reductions** ($\overset{\cdot\cdot}{\rightarrow}$) and **graph reductions in finite positive computer time** ($\overset{+}{\rightarrow}$) as seen in:



Parse tree Syntax tree

Example 3. Macros are defined using the directive $[\ddot{=}]^{\cdot}$. New **operators** may be defined using one of the directives $[\doteq]^{\cdot}$ or $[\acute{=}]^{\cdot}$. The difference is just that the latter directive forces Map to check the computer hardware to see whether the operator has been hardware implemented already, and then use this (fast) implementation instead. Many computers have an increment facility implemented in their hardware, and therefore the Mac operator $[x^{+}]^{\cdot}$ has been defined as $x^{+} \acute{=} x + 1$. On page 626 we see that this operator is denoted $\boxed{u^{+}}$ in parse trees (and syntax trees!).

Substitution of equals

Computing the value 14 by reductions of the syntax tree above, we made use of the fact that the expressions $[3 + 4]^{\cdot}$ and $[7]^{\cdot}$ are equal in the Mac System and thus may substitute each other in any expression. Since syntax trees contain the parenthesis structure of a term **implicitly**, we did not think of **parenthesis** when doing the substitution, but we are forced to that when substituting equals in the ordinary (compact) notation.

Example. $[2 \cdot 3]^{\cdot}$ and $[10 - 4]^{\cdot}$ are equals. Substituting the symbols $2{\cdot}3$ in $[2{\cdot}3^{+}]^{\cdot}$ we obtain $[10\text{-}4^{+}]^{\cdot}$, but the values of these two expressions are 8 and 5, respectively!

Correct substitution of equals

1) Is there a visible or a hidden parenthesis around the expression you want to substitute?

- Examples: $[-2{\cdot}3]^{\cdot}$ has a hidden parenthesis around $2{\cdot}3$, since $[-2{\cdot}3 \equiv -(2{\cdot}3)\ ]$, whereas $[2{\cdot}3^{+} \equiv (2{\cdot}3)^{+}]^{\cdot}$ does not hold.

2) If so, you may substitute the expression, **but** there must be a parenthesis around the new subexpression unless the parenthesis is superfluous!

- Example: $[-2{\cdot}3]^{\cdot}$ has a hidden parenthesis around $2{\cdot}3$, and $2{\cdot}3$ is equal to 10 - 4.
  Substitution of equals give $[-(10 - 4)]^{\cdot}$
  and here the parenthesis is necessary for correct substitution!