Interactive Transformations of Expressions

Rule−based webMathematica applications

Bernhard Zgraggen

zgraggen@sis.unibe.ch
Hochschule für Technik Rapperswil HSR, Switzerland

Interactive rule−based transformations of expressions often play a central role in mathematical Distance Education projects.
In such projects interactive applications are programmed on the basis of a Java−Interface to *Mathematica* allowing the users to solve certain computations step−by−step with interactive hints and electronic assistance.
Users of these applications may browse forward and backward through several computations, may propose solutions between the single steps, may administrate and work on several computations in parallel and continue working on any computation.
The paper demonstrates some of the applications and explains the main constituents of the underlying *Mathematica* code design by a distinction of model and view functionalities. Several levels of interactivity are discussed and illustrated with examples.

■ Basic Examples − towards interactivity

In this section parts of a web−project (called *webSolutions*) representing approaches with restricted interactivity are illustrated and described with the aid of a continuous example. On the one hand these approaches serve as a starting point for clearing up the needs of enhanced interactivity, on the other hand they demonstrate that different searching strategies with distinct levels of interactivity and view−components may be developped upon a core basis of transformation rules.

□ Detailed Step−by−Step−Solutions

Computing a limit of a function, as $\lim_{u\to\infty} \frac{\text{Log}[u]}{u}$, may be implemented with the aid of a greedy searching strategy with depth−limitations for Bernouilli−Hospital−rules transforming expressions according to a set of rules. The next cell shows the output when a solution for this mathematical problem is requested. The request form allows inputs for the functional formula ($\frac{\text{Log}[u]}{u}$), the variable (**u**), the point to be approached (∞), the depth−limitation (**2**) for iterative applications of Bernouilli−Hospital rules, the language for explanations ("English") and the font size ("12") for formulas. In addition, the corresponding web−interface offers to convert the HTML output (Hypertext MarkUp Language) with date, time and list of parameters to a single graphics (.gif). This conversion and the parametrization of the request form constitute the essential possibilites of interactions in this approach.

Example 1: Dynamic step−by−step solution output to a limit problem with the aid of a greedy searching strategy (from the subproject *DetailedSolutions*)

The system concludes that the limit exists and can be computed as follows:

$$\lim_{u \to \infty} \left(\frac{\text{Log}[u]}{u}\right)$$

HospitalInfinityRule (∞/∞): $\lim_{u \to \infty} \left(\frac{\text{Log}[u]}{u}\right) = \lim_{u \to \infty} \left(\frac{1}{u}\right)$

$$= \lim_{u \to \infty} \left(\frac{1}{u}\right)$$

ReciprokeRule: $\lim_{u \to \infty} \left(\frac{1}{u}\right) = \frac{1}{\lim_{u \to \infty} (u)}$

$$= \frac{1}{\lim_{u \to \infty} (u)}$$

IdentityRule: $\frac{1}{\lim_{u \to \infty} (u)} = 0$

$$= 0$$

$$\boxed{0}$$

The example shown above belongs to the subproject *DetailedSolutions*. Similar dynamic applications were developped for mathematica subjects as linear systems of equations, linear optimization, optimization with constraints, analyzing functions by means of calculus, Taylor series expansion and the computation of (partial) derivatives [cf. 1, 2].

□ Generic Solutions

Many mathematical problems of a rather simple nature have a solution based on a systematic application of transformation rules. A searching strategy like the greedy search illustrated above in example 1 leads to unsatisfactory results in many situations. A heuristic best−first search often is a much better and more flexible strategy. Such a strategy is the basis of the subproject *GenericSolutions* [cf. 2, 3].
Let us widen the first example to the following computational problem of finding $\mathbf{lim}_{x\to\infty}$ **ArcTan[Sin[x]]**$\frac{\mathrm{Log}[x]}{x}$. It may be solved with the aid of an extended set of transformation rules combined with a heuristic best−first search strategy.

The next cell shows the output of a request for the mathematical problem mentioned above. The request form provides input fields for the function formula (**ArcTan[Sin[*x*]]** $\frac{\mathrm{Log}[x]}{x}$), the variable (**x**) and the point to be approached (∞).
Additionally, the search algorithm may be parametrized on a meta level: The design of the algorithm provides parameters for the time, depth, breadth and total number of nodes in the search tree, parameters for the mathematical context (transformation rules, pre−functions, post−functions, complexity functions, heuristics, proposed solutions), the notational context (formatting rules) and the style (colour, font, fontsize, face, weight etc.) of the output.
As before, the corresponding web−interface offers to convert the HTML output (Hypertext MarkUp Language) with date, time and list of parameters to a single graphics (.gif). This conversion and the parametrization of the request form constitute the essential possibilites of interactions in this approach.

Example 2: Finding a step−by−step solution to a limit problem with the aid of a generic and heuristic best−first search algorithm (from the subproject *GenericSolutions*)

The following initial expression was entered:

$$\lim_{x\to\infty}\left(\frac{\mathrm{ArcTan}[\mathrm{Sin}[x]]\,\mathrm{Log}[x]}{x}\right)$$

The *Mathematica* system proposes the following solution to your problem:

$$\mathrm{Limit}\left[\frac{\mathrm{ArcTan}[\mathrm{Sin}[x]]\,\mathrm{Log}[x]}{x},\ x\to\infty\right]$$

The *Step-by-Step Solver* terminated searching with the following state(s):

Expression found fulfilling conditions for solutions.

The *Step-by-Step Solver* found the following optimal solution by applying 6 transformation rules:

0

Here are the 6 steps leading to the optimal solution:

Product Rule: The limit of the product *fg* is equal to the product of the limes of *f* by the limes of *g* provided the limits exist and their product is not indeterminate (cases as 0∗∞ or ∞∗0 are not covered).

$$\lim_{x\to\infty}\left(\frac{\text{ArcTan}[\text{Sin}[x]]\,\text{Log}[x]}{x}\right) =$$

$$\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])\,\lim_{x\to\infty}\left(\frac{\text{Log}[x]}{x}\right)$$

Hospital Infinity Rule 1: The limit of a quotient $\frac{f}{g}$ is equal to the limit of the quotient of derivatives $\frac{f'}{g'}$ if the limits of f and g are infinite and the derivatives and the limit of their quotient exist.

$$\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])\,\lim_{x\to\infty}\left(\frac{\text{Log}[x]}{x}\right) =$$

$$\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])\,\lim_{x\to\infty}\left(\frac{1}{x}\right)$$

General Power Rule: The limit of the power f^a is equal to the limit of f by a if a is constant and the limit of f exists.

$$\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])\,\lim_{x\to\infty}\left(\frac{1}{x}\right) =$$

$$\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])\,\frac{1}{\lim_{x\to\infty}(x)}$$

Identity Rule: $\lim_{x\to x_0}(x) = x_0$

$$\frac{\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])}{\lim_{x\to\infty}(x)} = \frac{\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]])}{\infty}$$

Interval Chain Rule 1: The inner function in the compound function oscillates in an interval

$$0\lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]]) = 0\,\text{ArcTan}\left[\lim_{x\to\infty}(\text{Sin}[x])\right]$$

Special Interval Rule: The function oscillates in an interval.

$$0\,\text{ArcTan}\left[\lim_{x\to\infty}(\text{Sin}[x])\right] = 0$$

An interesting point in connection with this example is the fact that *Mathematica* fails to compute the limit with its analytic methods because of an essential singularity of the function (recognizable when a series expansion is tried):

```
Limit[(Log[x] / x) ArcTan[Sin[x]], x → +Infinity]
```

$$\text{Limit}\left[\frac{\text{ArcTan}[\text{Sin}[x]]\ \text{Log}[x]}{x}, x \to \infty\right]$$

Such problems may be rectified with the aid of generic product rules as:

```
{{lim[f_ g_, x_, x0_] :> lim[f, x, x0] lim[g, x, x0] /;
    Hold[
     (! FreeQ[f, x] && ! FreeQ[g, x] &&
       limitRealQ[f, x, x0] && limitRealQ[g, x, x0] &&
       Limit[f, x → x0] Limit[g, x → x0] =!= Indeterminate
     )
    ]
 }, {}, {"Product Rule"},
 {"The limit of the product fg is equal to the product of the
    limes of f by the limes of g provided the limits exist
    and their product is not indeterminate (cases as 0*∞
    or ∞*0 are not covered)"}, {" lim (f g) = lim f  lim g"}}
                                x→x0      x→x0   x→x0
```

The utility–function `limitRealQ[f, x, x0]` checks whether the real limit of the function `f` exists if x approaches the point `x0` using *Mathematicas* built–in `Limit` function.

```
{{lim[f_ g_, x_, x0_] :> lim[f, x, x0] * lim[g, x, x0] /;
    Hold[
     ! FreeQ[f, x] && ! FreeQ[g, x] &&
      (intervalFiniteRealQ[f, x, x0] && limitRealQ[g, x, x0] &&
        (Limit[g, x → x0] === 0 || Limit[g, x → x0] === 0.)
      )
    ]
 }, {}, {"Interval Times Zero Limit Rule"},
 {"The limit of a product of two expressions exists and is
    equal to 0 if the first expression is locally bounded
    and the limit of the second expression is equal to 0."},
 {" lim (f g) = [a, b]×0"}}
   x→x0
```

The utility–function `intervalFiniteRealQ[f, x, x0]` checks whether the function `f` is locally bounded around the point `x0` using *Mathematicas* built–in `Limit` function.
The structure and components of such rules are described below.

■ Levels of interactivity

□ Input–Output–Level

In the first example above, essentially, interactions are restricted to the input of mathematical parameters as the function formula ($\frac{\text{Log}[u]}{u}$), the variable (**u**) and the point to be approached (∞): The parametrization of the language of explanations ("English") and the font size ("12") for formulas concern the appearance of the output.

The parameter for the depth–limitation (**2**) for iterative applications of Bernouilli–Hospital rules actually is a meta parameter controlling the behaviour of the search procedure..

□ Meta−Level

The parameters in the meta−level allow to control the search strategy. Many meta−parameters may be found in the context of a heuristic best−first search algorithm using generic tranformation rules: The design of the algorithm provides parameters for the time, depth, breadth and total number of nodes in the search tree, parameters for the mathematical context (transformation rules, pre−functions, post−functions, complexity functions, heuristics, proposed solutions), the notational context (formatting rules) and the style (colour, font, fontsize, face, weight etc.) of the output. These parameters and the algorithm design of the generic heuristic best−first search are desribed in detail in [3].

□ Solution level

By interactions in the solution level we describe operations as:

- Selecting transformation rules, input of a position where the selected rule may be applied in the current expression and input of the resulting tranformation.
- Backtracking through the steps of the solution or recomputing the solution from an arbitrary step (as a starting point).
- Getting hints for the separate steps in the solution, and.
- Administrating and working on several computations simultaneously.

In the subprobject *ActiveSolutions* applications have been developped that allow such interactions during the expansion of the solution. These applications are described and illustrated in [2, 4].

The figure below is screen shot in the run of an interactive application. It shows a situation where an adequate transformation rule has to be selected. If the users selection does not match with the form of the current expression a hint link may be followed.

Figure 1: Screenshot from the subproject *ActiveSolutions*: A hint is provided if the selected rule does not match the form of the current expression. The hint contains a list of all matching rules.

■ The *Mathematica* code framework

In this chapter we present fragmentary descriptions of the most important constituents of the *Mathematica* code framework that forms the basis of the *ActiveSolutions* project. The presentation is structured by distinguishing between a logical and a view level. It is not always possible to make such a distinction sharply. Therefore we give cross−references to the view level, occasionally.

□ The logical level

Initialization

We resume the second example above concentrating on the last two steps of the solution:

```
Interval Chain Rule 1: The inner function in
the compound function oscillates in an interval
```

$$0 \lim_{x\to\infty}(\text{ArcTan}[\text{Sin}[x]]) = 0\,\text{ArcTan}\left[\lim_{x\to\infty}(\text{Sin}[x])\right]$$

```
Special Interval Rule: The
function oscillates in an interval.
```

$$0\,\text{ArcTan}\left[\lim_{x\to\infty}(\text{Sin}[x])\right] = 0$$

It is easy to see that the treatment of a binary product containing a zero factor depends of the form of the nonzero factor. The zero factor does not absorb the nonzero factor if the latter contains parts with limits to be solved. This logic is a part of the inititialization:

```
Unprotect[Times];
Times[x_, 0] := x MyZero /; MemberQ[x,
   lim[___] | lim | "lim" | "ℓim", {0, Infinity}, Heads → True];
Times[x_, MyZero] := x 0 /; ! MemberQ[x,
    lim[___] | lim | "lim" | "ℓim", {0, Infinity}, Heads → True];
Protect[Times];
```

The display of the constant `MyZero` is controlled by a set of rules for formatting expressions (`ExpressionFormattingRules`); actually, they belong to the view level. Sums containing -∞ or +∞ are treated in an analogous way.

Parsing

Parsing concerns input values for the function, the variable and the point to be approached (problem parameters). Since there are three parameters there are different combinations of them that may be critical. Every parameter may be parsed separately or in combination with the others with respect to syntactic or semantic errors. In the example of computing limits parsing obeys the following principles:

- The variable must be a non–numeric symbol and it must not be an element of `{I,E,Pi,Catalan,GoldenRatio,EulerGamma,°,Infinity}`. Moreover, it must not appear as a head of any functional term. Inputs, e.g., resulting in the problem expression `lim[f[x], f, x]` (meaning $\lim_{f\to x}$ `f[x])` `yield` would lead to a parsing error.
- The input expression for the function must not have the symbol `lim` as a functional head of any of its partial expressions, because this symbol is reserved and has the meaning of a limit operator that is used in the left hand sides of the transformation rules and thus plays a crucial role in rule matching.

Generating and comparing expressions

There is a function `createFirstExpression` generating a functional initial expression from the input parameters with the aid of the operator `lim` as, e.g., `lim[f[x], x, x0]from {f[x], x, x0}. The form with the operator lim is compatible with the left hand sides of the rules.`

Another typical function, `compareExpressions`, is used when partial results proposed by the user have to be checked for correctness.

```
compareExpressions[expr1_, expr2_] :=
 FullSimplify[expr1] === FullSimplify[expr2]
```

Simplifications or transformations that are admissible in a certain mathematical context must not influence the evaluation concerning correctness of the proposed result. As a matter of course, commuting the factors in the second step of the second example above, e.g., still leads to a correct result.

Mathematical transformation rules

Transformation rules representing a mathematical context are central to the code framework of the applications in the subprojects *GenericSolutions* and *ActiveSolutions* [2]. In the application for computing limits there are more than 40 transformation rules. The examples below show variations of the product well–known in the area of computing limits. Indeterminate expressions as 0∞ are covered by special rules as Bernouilli–Hospital, Taylor series expansion or specific algebraic transformations. The rules consist of five components that are explained in detail in [3]. The last two components may be regarded as belonging to the view level because they serve for displaying explanations.

Product rule Version 1: This is the classical product rule $\lim_{x \to x_0} (f\ g) = \lim_{x \to x_0} f \lim_{x \to x_0} g$: A determinate product of two limits is equal to the limit of the product. The utitiliy–function `limitRealQ` serves to decide whether the real limit of a function exists. This function is essentially based on the analytical methods used by *Mathematica*s built–in `Limit` function.

```
{{lim[f_ g_, x_, x0_] :> lim[f, x, x0] lim[g, x, x0] /;
    Hold[
     (! FreeQ[f, x] && ! FreeQ[g, x] &&
       limitRealQ[f, x, x0] && limitRealQ[g, x, x0] &&
       Limit[f, x → x0] Limit[g, x → x0] =!= Indeterminate
     )
    ]
 }, {}, {"Product Rule"},
 {"The limit of the product fg is equal to the product of the
    limes of f by the limes of g provided the limits exist
    and their product is not indeterminate (cases as 0*∞
    or ∞*0 are not covered)"}, {" lim (f g) = lim f  lim g"}}
```

Product rule Version 2: This is a rule of type Bernouilli–Hospital: $\lim_{x \to x_0} (f\ g) = \lim_{x \to x_0} \left(\frac{-f' g^2}{g'} \right)$: An indeterminate product 0∞ of limits may be computed by derivatives. The extensive conditions and utility–functions optimize the application of this rule by avoiding transformations that result in expressions getting more and more complex.

```
{{lim[f_ g_, x_, x0_] :>
   lim[Hold[Simplify[-D[f, x] g^2 / D[g, x]]], x, x0] /;
    Hold[
     Not[FreeQ[f, x]] &&
      Not[FreeQ[g, x]] && limitRealQ[f, x, x0] &&
      (Limit[f, x -> x0] === 0 || Limit[f, x -> x0] === 0.) &&
      limitRealQ[g, x, x0] &&
      Head[Limit[g, x -> x0]] === DirectedInfinity
    ]
 },
 {Hold[
   ((LeafCount[
        reduceExpression[Simplify[-D[g, x] f^2 / D[f, x]], x]] >
       LeafCount[reduceExpression[
         Simplify[-D[f, x] g^2 / D[g, x]], x]] ||
      recursiveAddPowerExponents[reduceExpression[
         Simplify[-D[g, x] f^2 / D[f, x]], x], x] >
       recursiveAddPowerExponents[reduceExpression[
         Simplify[-D[f, x] g^2 / D[g, x]], x], x] ||
      (-D[f, x] g^2 / D[g, x] /. {x -> x0}) =!= Indeterminate)
   )
  ]
 }, {"Hospital Zero Rule 2"},
 {StringForm["The limit of a product f g is equal to the limit
     of the quotient of derivatives '1' if the limit of
     f is equal to 0, the limit of g is infinite and the
     derivatives and the limit of the quotient '1' exist.",
   StandardForm["-f' g^2 / g'"]]}, {" lim_{x->x0} (f g) = lim_{x->x0} (-f' g^2 / g') "}}
```

Product rule Version 3: $\lim_{x \to x_0} (uv) = \lim_{x \to x_0} \left(\frac{1}{4} (u + v)^2 - \frac{1}{4} (u - v)^2 \right)$: An indeterminate product of the form $\infty\, 0$ of two limits is computable by an algebraic transformation.

Again, the extensive conditions and utility−functions optimize the application of this rule by avoiding transformations that result in expressions getting more and more complex.

```
{{lim[u_ v_, x_, x0_] :>
   lim[(1 / 4) (u + v)^2 - (1 / 4) (u - v)^2, x, x0] /;
    Hold[
     Not[FreeQ[u, x]] && Not[FreeQ[v, x]] &&
      (Head[Limit[u, x → x0]] === DirectedInfinity &&
        limitRealQ[u, x, x0] && limitRealQ[v, x, x0] &&
        (Limit[v, x → x0] === 0 || Limit[v, x → x0] === 0.)
      )
    ]
 },
 {Hold[
   Not[
    Or[
     (
      LeafCount[
         reduceExpression[Simplify[-D[v, x] u^2 / D[u, x]], x]] >
       LeafCount[reduceExpression[Simplify[
          -D[u, x] v^2 / D[v, x]], x]] ||
        recursiveAddPowerExponents[reduceExpression[
          Simplify[-D[v, x] u^2 / D[u, x]], x], x] >
        recursiveAddPowerExponents[reduceExpression[
          Simplify[-D[u, x] v^2 / D[v, x]], x], x] ||
        (-D[u, x] v^2 / D[v, x] /. {x → x0}) =!= Indeterminate
     )
     ,
     (
      LeafCount[
         reduceExpression[Simplify[-D[u, x] v^2 / D[v, x]], x]] >
       LeafCount[reduceExpression[Simplify[
          -D[v, x] u^2 / D[u, x]], x]] ||
        recursiveAddPowerExponents[reduceExpression[
          Simplify[-D[u, x] v^2 / D[v, x]], x], x] >
        recursiveAddPowerExponents[reduceExpression[
          Simplify[-D[v, x] u^2 / D[u, x]], x], x] ||
        (-D[v, x] u^2 / D[u, x] /. {x → x0}) =!= Indeterminate
     )
    ]
   ]
  ]},
 {"Infinity Times Zero Special Rule"},
 {StringForm["The limit of u v is equal to the limit
     of '1' if the latter one exists and the limits
     of u and v are infinite or 0, respectively.",
   StandardForm["1/4 (u + v)^2 - 1/4 (u - v)^2"]]},
 {"lim_{x→x0} (u v) = lim_{x→x0} (1/4 (u + v)^2 - 1/4 (u - v)^2"}}
```

Beside the transformation rules representing the mathematical context there are rules controlling the display of expressions (`ExpressionFormattingRules`) and rules (`RuleFormattingRules`). They may be regarded as belonging to the view level because their purpose concenrs to display expressions and explanations.

Complete logic of a computational step

There is a fundamental function in the logic level describing the complete logic of a computational step. This function returns a list containing all rules with corresponding positions and transformations that match with an expression. All checks of results as rule selection, position input and transformations as well as all hints are derived from this list. In the case of the limit problem $\lim_{x \to \infty} \left(\frac{\text{ArcTan}[\text{Sin}[x]]\,\text{Log}[x]}{x} \right)$ there arise five rules that are applicable at the top position {} of the expression. The applications of these rules lead to the expressions given in the corresponding sublists.

$$\{\{\text{HospitalInfinityRule2},\ \{\{\lim\left[-\frac{\text{ArcTan}[\text{Sin}[x]]^2\ \left(1+\text{Sin}[x]^2\right)}{x\left(x\,\text{Cos}[x]-\text{ArcTan}[\text{Sin}[x]]\ \left(1+\text{Sin}[x]^2\right)\right)},\ x,\ \infty\right],\ \{\}\}\}\},\ \{\text{HospitalInfinityRule3},\ \{\{\lim\left[-\frac{\text{ArcTan}[\text{Sin}[x]]^2\ \left(1+\text{Sin}[x]^2\right)}{x\left(x\,\text{Cos}[x]-\text{ArcTan}[\text{Sin}[x]]\ \left(1+\text{Sin}[x]^2\right)\right)},\ x,\ \infty\right],\ \{\}\}\}\},\ \{\text{HospitalZeroRule2},\ \{\{\lim\left[\left(\frac{\text{ArcTan}[\text{Sin}[x]]}{x}+\frac{2\,\text{Cos}[x]}{-3+\text{Cos}[2\,x]}\right)\text{Log}[x]^2,\ x,\ \infty\right],\ \{\}\}\}\},$$

$$\{\text{HospitalZeroRule3},\ \{\{\lim\left[\left(\frac{\text{ArcTan}[\text{Sin}[x]]}{x}+\frac{2\,\text{Cos}[x]}{-3+\text{Cos}[2\,x]}\right)\text{Log}[x]^2,\ x,\ \infty\right],\ \{\}\}\}\},\ \{\text{ProductRule},\ \{\{\lim[\text{ArcTan}[\text{Sin}[x]],\ x,\ \infty]\ \lim\left[\frac{\text{Log}[x]}{x},\ x,\ \infty\right],\ \{\}\}\}\}\}$$

Error detection and hints

There are error detection and hint functions to all the essential mathematical interactions. They are based on the complete compuational step described in the subsection above.

- Rule selection: All rules that match with the current working expression are provided as a list whenever a non−matching rule is selected.
- Position input: There are two kinds of hints concerning positions. On the one hand the current working expression may be displayed in a tree form with positions added to all subexpressions. On the other hand all positions at which a specific matching rule is applicable may be displayed as a list.
  The first kind of hint is illustrated in Figure 2 below. Such a hint is provided when the user proposes a non−existing position..

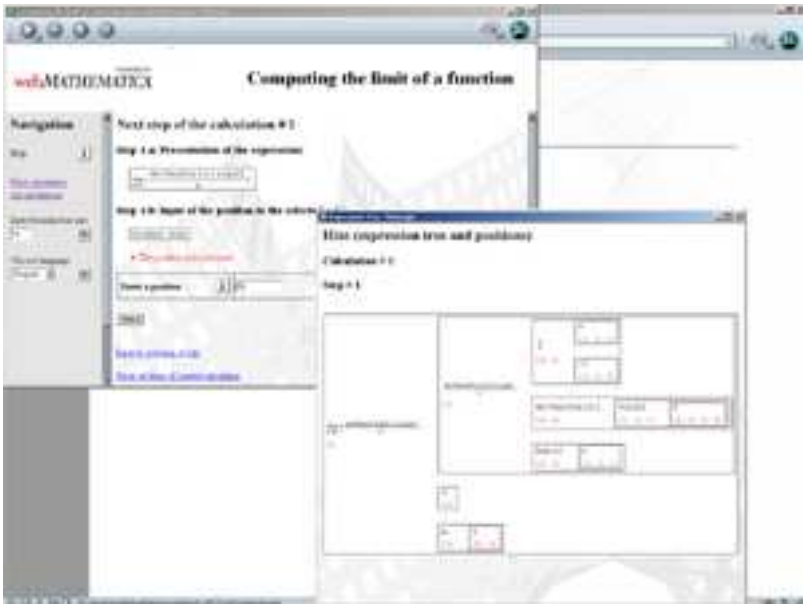


Figure 2: *ActiveSolutions*: After entering a non−existing position a hint is provided displaying a tree form of the current expression with positions added to all subexpressions.

- Expression input: By the selected rule and the associated position input the transformed expression for the current computational step is determined. It may be received as a hint. An expression is considered as erroneous if its fully simplified version (`FullSimplify`) is not identical (`SameQ`) with the fully simplified version of the correct transformed expression. Such a comparison is done by the logical function `compareExpression` described above.

Figure 3: *ActiveSolutions*: After proposing an erroneously transformed expression a hint is provided displaying a correct expression.

Mathematical hints as presented above are provided only if erroneous propositions for rules, positions or transformed expressions are made. For this reason error detecting functions are important parts of the logical level. Such functions are based on the complete logic of a computational step.

End of computation

A computation is regarded as finished if the transformed expression does not contain the `lim`–operator as a functional head.

```
calculationFinishedQ[newexpression_] :=
  FreeQ[newexpression, lim[___], Heads → True]]
```

Application control

The application in ActiveSolutions are controlled by Javs Server Pages (JSP). These pages provide the following functionalities:

- At any time a new computation may be started and several computations may be worked on simultaneously.
- All administrations may be administrated: Computations may be removed or details of computations may be displayed.
- In the course of a computation backtracking operations may be executed within a computational step. It is also possible to resume a computation at any step of the expansion as a new starting point.

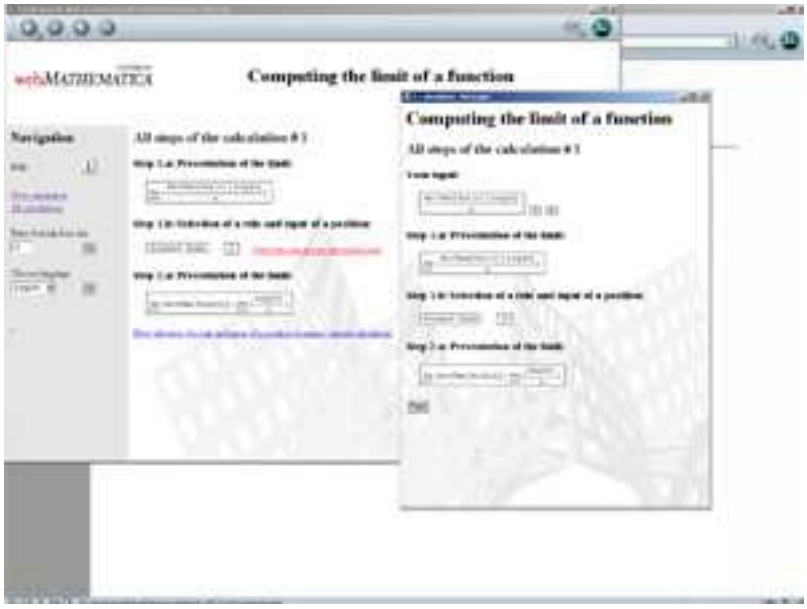


Figure 4: *ActiveSolutions*: All steps in the course of a computation may be displayed and it is possible to resume a computation at any computational step.

Generic versus specific functions

The functions in the logical level may be divided into generic ones and ones that are specific for the mathematical application. Here "generic" means the availability of a flexible interface allowing to parametrize the function with specific values as set of rules representing different mathematical contexts.

Generic are the functions controlling the application, computing hints, computing a complete logic of a step and generating expression trees with sub−positions. The logic of initialization, the set of transformation rules and formatting rules, the creation of expressions, the comparison of expressions, the logic of finishing a computation and the parsing functions are specific for the mathematical context.

□ View level

The numerous functionalities belonging to the view level are rather diverse as it is to be expected for an interactive web application of the kind described here. Such functionalities include the HTML (Hyper text markup language) display in a browser as well as technologies such as Java Server Pages (JSP) and web*Mathematica*. Below only interesting functions of the view level are described that are in close connection with *Mathematica* and that are particular for the applications of *webSolutions* [2].

Explanational parts of transformation rules

Mathematical transformation rules contain two components related to explanatoins. The first one is an explanational text displayed when a rule is applied in the course of a computation (cf. Example 1 and 2 above). The second one is a formula describing the rule compactly as an icon; it is used when a rule has to be selected (cf. Figure 1 above). The code fragment below highlights the two explanational components of the constant rule.

```
{{lim[c_, x_, x0_] :> c /; Hold[FreeQ[c, x]]},
 {}, {"Constant Rule"},
 {"The limit of a constant expression is equal
    to the constant.", {" lim (c) = c"}}}
                      x→x0
```

Rules for displaying expressions

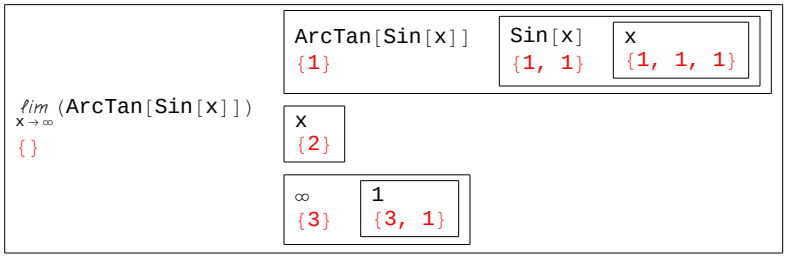
A specific set of rules controls the display of expressions, particularly the display of the limit operator, of intervals and much more. The example below makes the role of such rules clearer. It shows the display of the expression `lim[ArcTan[Sin[x]]Log[x]/x, x, Infinity] + Interval[{a,b}]`:

$$[a,b] + \lim_{x \to \infty} \left(\frac{\text{ArcTan}[\text{Sin}[x]]\ \text{Log}[x]}{x} \right)$$

Display of expressions as trees with positions added to subexpressions

The logical functions computing hints for the system of positions of an expression (cf. Figure 2) are related to corresponding functions in the view level. The system of positions is displayed as tree–like table collecting all subexpressions with associated positions. The corresponding functions are generic (contrary to the view level functions in the two subsections above).
The example below illustrates the system of positions for the expression `ArcTan[Sin[x]]`.

Example 3: Generic display of the system of position and subexpressions for the expression `ArcTan[Sin[x]]`.



■ References

[1] B. Zgraggen, "*Interactive and Dynamic Step−By−Step−Solutions to Mathematical Problems in the World Wide Web*", Proceedings of the 5th International Conference on New Educational Environments, Lucerne, 2003.

[2] *webSolutions* Homepage: http://www.webmath.ch

[3] B. Zgraggen, "*Design and Applications of a Generic and Heuristic Step−By−Step− Problem Solver*". Proceedings of the 6th International *Mathematica* Symposium, Banff (Canada), 2004, Wolfram Technology Conference, Champaign, 2004, http://library.wolfram.com/infocenter/conferences/5421/.

[4] B. Zgraggen, "*Interactive Transformations of Expressions*", Wolfram Technology Conference, Champaign, 2005, http://library.wolfram.com/infocenter/conferences/5765/.