The ribbonproofs package

John Wickerson

Version 1.0 – 26 June 2013

Abstract

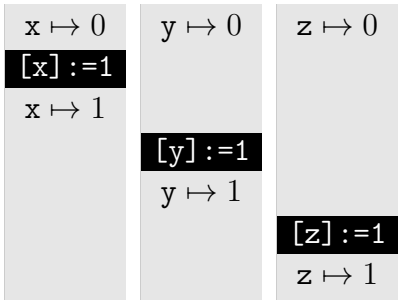
The ribbonproofs package provides a mechanism for drawing ribbon proofs in LaTeX. Ribbon proofs form a diagrammatic proof system, due to Wickerson, Dodds and Parkinson [4], for *separation logic* [3]. They build on an earlier system, due to Bean [1], for *bunched implications logic* [2].

Contents

1 Getting started

1.1 Installation

Place `ribbonproofs.sty` into the same directory as the `.tex` file you are compiling. To test that the package is working correctly, you might try to compile the little document given in Fig. 1. The code is given on the lefthand side, with the expected output to the right.




```
\documentclass{standalone}
\usepackage{ribbonproofs}
\begin{document}
\def\to{\mapsto}
\begin{ribbonproof}[start ribbons={
  a/{left=0, right=16, label={\tt x}\to0},
  b/{left=18, right=34, label={\tt y}\to0},
  c/{left=36, right=52, label={\tt z}\to0}}]
\com[finish ribbons={a}, start ribbons={
  a/{label={\tt x}\to1}}]{[x]:=1} \\
\com[finish ribbons={b}, start ribbons={
  b/{label={\tt y}\to1}}]{[y]:=1} \\
\com[finish ribbons={c}, start ribbons={
  c/{label={\tt z}\to1}}]{[z]:=1} \\
\end{ribbonproof}
\end{document}
```

Figure 1: A test document and its expected result

1.2 A simple example

Figure 1 provides a good starting point for examining how the ribbonproofs package works.

All the action takes place within the `ribbonproof` environment. Upon entering this environment, we provide the `start ribbons` key, in order to introduce some ribbons at the top of the diagram. We name them `a`, `b` and `c`, give the horizontal positions of their `left` and `right` edges, and provide a `label`.

We then proceed to build up the diagram in a series of rows, each terminated with `\\`. In this example, each row comprises a single `\com` step (`\com` stands for 'command'). Each `\com` has a number of optional parameters. The `finish ribbons` key lists those ribbons that end at this step; they form the step's precondition. The `start ribbons` key defines ribbons that begin at this step; they form the step's postcondition. Note that we used the same syntax (`start ribbons`) to introduce some ribbons at the beginning of the proof. Each `\com` has one mandatory parameter: the text, which is printed in white on the step.

2 Examples

2.1 List append

Figure 2 shows the result.

```
\begin{ribbonproof}[start ribbons={
  c/{left=35,right=53,label=\ls\,\x\,0},
  e/{left=74,right=86,label=\ls\,\y\,0}}]
\startblock[extra left=33, extra right=2,
  fit ribbons={c,e}, start ribbons={
    d/{left=\RIBcRIGHT+2,right=\RIBeLEFT-4,label=\x\doteq 0}
  }]{if (x==0) \{}
\\
```

```
\jus[finish ribbons={c,d}]{Unfold $\ls$ def}
\com[finish ribbons={e}, start ribbons={
  e/{label=\ls\,\x\,0}}]{x:=y}
\\[-6]
\moveribbons{e/{left=4}}
\\[-1]
\continueblock[repeat labels, start ribbons={
  d/{label=\x\dotneq 0}}]{\} else \{}
\\[-1]
\jus[start ribbons={
  a/{left=6,right=\RIBcLEFT-2,label=\ls\,\x\,\x}}]{Fold $\ls$ def}
\\[-4]
\moveribbons{c/{left=20},a/{right=18}}
\\[-4]
\com[finish ribbons={a,c,d}, start ribbons={
  a/{label=\ls\,\x\,\t},
  c/{label=\ls\,\t\,0},
  d/{label=\t\dotneq 0}}]{t:=x}
\\
\jus[finish ribbons={c,d}, start ribbons={
  c/{label=\exists U\ldotp\t\mapsto U * \ls\,U\,0}}]{Unfold $\ls$ def}
\\
\com[finish ribbons={c}, start ribbons={
  b/{left=\RIBcLEFT,right=33,label=\t\mapsto\u},
  c/{left=35,right=\RIBcRIGHT,label=\ls\,\u\,0}}]{u:=[t]}
\\
\startblock[fit ribbons={a},
  extra left=2, extra right=2,start ribbons={
  d/{label=\u\dotneq 0}}]{while (u!=0) \{}
\\
\jus[finish ribbons={a,b,c}, height=10, start ribbons={
  a/{right=\RIBbRIGHT, label=\ls\,\x\,\u},
  c/{label=\ls\,\u\,0}}]{Lemma: $\ls\,a\,b *\ls\,b\,c * \ls\,c\,0$ \\
    implies $\ls\,a\,c * \ls\,c\,0$}
\\
\moveribbons{a/{right=18},c/{left=20}}
\\[-3]
\com[finish ribbons={a,c,d}, start ribbons={
  a/{label=\ls\,\x\,\t},
  c/{label=\ls\,\t\,0},
  d/{label=\t\dotneq 0}}]{t:=u}
\\
\jus[finish ribbons={c,d}, start ribbons={
  c/{label=\exists U\ldotp\t\mapsto U * \ls\,U\,0}}]{Unfold $\ls$ def}
\\
\com[finish ribbons={c}, start ribbons={
```

```
  b/{left=\RIBcLEFT,right=33,label=\t\mapsto\u},
  c/{left=35,right=\RIBcRIGHT,label=\ls\,\u\,0}}]{u:=[t]}
\\
\finishblock[start ribbons={
  d/{label=\u\doteq 0}}]{\}}
\\
\jus[finish ribbons={b,c,d}, start ribbons={
  b/{label=\t\mapsto0}}]{Unfold $\ls$ def}
\\
\com[finish ribbons={b}, start ribbons={
  b/{label=\t\mapsto\y}}]{[t]:=y}
\\
\jus[finish ribbons={b,e}, start ribbons={
  b/{right=\RIBeRIGHT,label=\ls\,\t\,0}}]{Fold $\ls$ def}
\\
\jus[finish ribbons={a,b}, start ribbons={
  a/{right=\RIBeRIGHT,label=\ls\,\x\,0}}]{
  Lemma: $\ls\,a\,b * \ls\,b\,0$ implies $\ls\,a\,0$}
\\
\finishblock{\}}
\\
\end{ribbonproof}
```

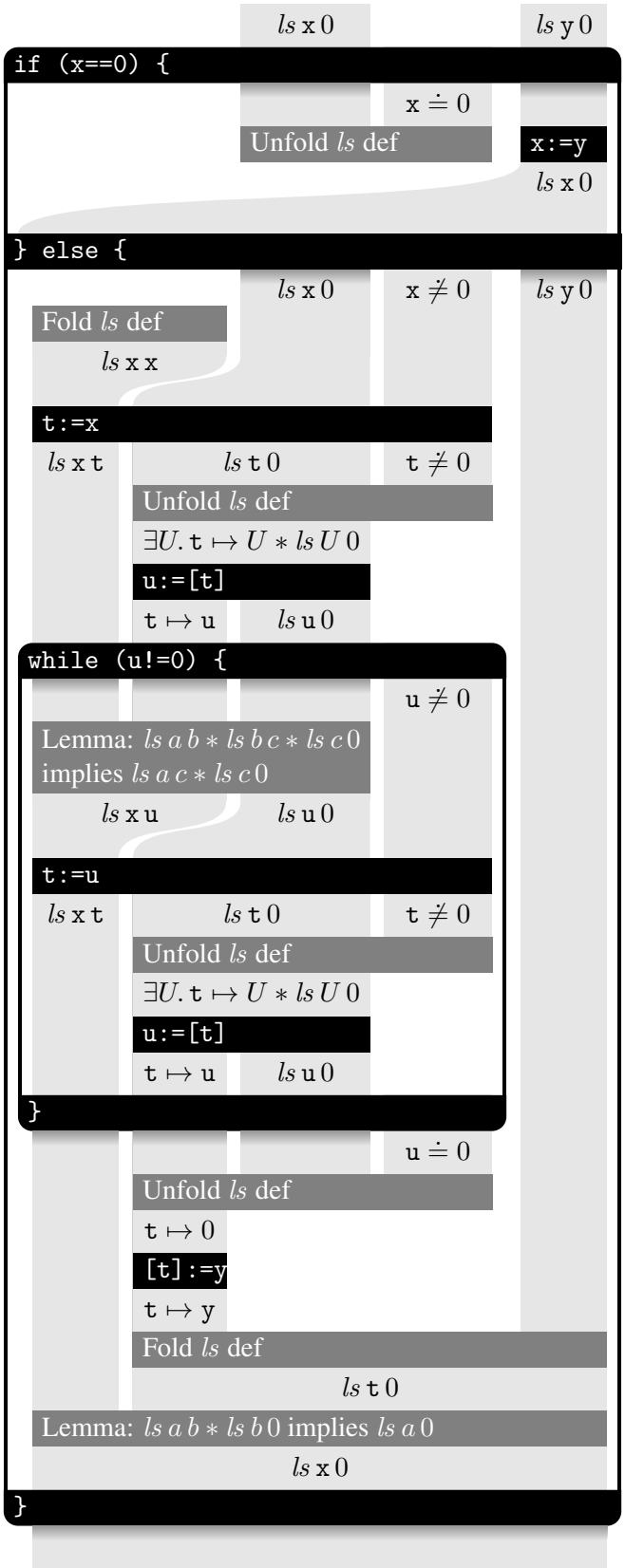


Figure 2: A ribbon proof of list append

2.2 List reverse

Figure 3 shows the result.

```
\begin{ribbonproof}[start ribbons={
  b/{left=20,right=38,label=\List\,\delta\,\x}}]
\com[start ribbons={
  c/{left=40,right=78,label=\List\,\varepsilon\,\y}}]{y:=0};
\\
\jus[finish ribbons={b,c}, start ribbons={
  b/{label=\List\,\alpha\,\x},
  d/{label=\delta\doteq \rev{\beta}\cdot\alpha,
    left=60, right=\RIBcRIGHT},
  c/{label=\List\,\beta\,\y, right=58}
}, start boxes={
  alpha/{left=14,right=82,label=\exists\alpha,color=SeaGreen},
  beta/{left=18,right=80,label=\exists\beta,color=MediumPurple}
}]{Choose $\alpha:=\delta$ and $\beta:=\varepsilon$}
\\
\startblock[fit boxes={alpha},
  extra left=10, extra right=2, start ribbons={
  a/{left=0,right=10,label=\x\dotneq 0}}]{while (x!=0) \{}
\\
\extendboxes{alpha/{left=-4}}
\\[-9]
\extendboxes{beta/{left=-8}}
\\[-9]
\jus[finish ribbons={a,b}, start ribbons={
  ab/{left=\RIBaLEFT, right=\RIBbRIGHT,
   label={\exists\alpha',i,Z\ldotp\x\mapsto i,Z \\
     {} * \List\,\alpha'\,Z * \alpha \doteq i \cdot \alpha'}}
  }]{Unfold $\List$ def}
\\[-11]
\moveribbons{
  c/{left=70,right=\RIBdRIGHT},
  d/{left=40,right=64}}
\moveboxes{
  alpha/{right=\RIBdRIGHT+4}}
\\[4]
\jus[finish boxes={alpha}, start boxes={
    alpha/{label=\exists\alpha},
    i/{left=\RIBabLEFT-2,right=\RIBdRIGHT+2,color=Chocolate,
      vertical offset=10,label=\exists i}
  }, finish ribbons={ab,d},
  start ribbons={
    ab/{label={\exists Z\ldotp\x \mapsto i,Z * \List\,\alpha\,Z}},
    d/{label=\delta\doteq \rev{\beta}\cdot(i\cdot\alpha)}
```

```
  }]{Choose $\alpha:=\alpha'$}
\\
\com[finish ribbons={ab}, start ribbons={
  a/{right=14,label=\List\,\alpha\,\z},
  b/{label={\x\mapsto i,\z}}
}]{z:=[x+1]}
\moveribbons{
  c/{left=\RIBdLEFT, right=50},
  d/{left=52, right=76}}
\moveboxes{
  alpha/{left=-8,right=82},
  beta/{left=-6},
  i/{right=78}}
\\
\com[finish ribbons={b}, start ribbons={
  b/{label={\x\mapsto i,\y}}}]{[x+1]:=y}
\jus[finish ribbons={d}, start ribbons={
  d/{label={\delta\doteq \rev{(i\cdot\beta)}\cdot\alpha}}}]
  {Reassociate $i$}
\moveboxes{
  i/{left=\RIBbLEFT-2},
  beta/{left=\RIBbLEFT-4}}
\\
\jus[finish ribbons={b,c}, start ribbons={
  c/{left=\RIBbLEFT,label={\List (i\cdot\beta)\,\x}}}]{Fold $\List$ def}
\\
\jus[finish ribbons={c,d}, finish boxes={i,beta},
  start boxes={
    beta/{vertical offset=10,horizontal offset=2,label=\exists\beta}},
  start ribbons={
    c/{label={\List\,\beta\,\x}},
    d/{label={\delta\doteq\rev{\beta}\cdot\alpha}}}]
  {Choose $\beta:=(i\cdot\beta)$}
\\[-6]
\moveribbons{
  d/{left=60,right=78},
  c/{left=40,right=58},
  a/{left=20,right=38}}
\moveboxes{
  alpha/{left=14},
  beta/{left=18}}
\\
\com[finish ribbons={c}, start ribbons={
  c/{label={\List\,\beta\,\y}}}]{y:=x}
\\[-5]
\com[finish ribbons={a}, start ribbons={
```

```
  b/{left=\RIBaLEFT,right=\RIBaRIGHT,label={\List\,\alpha\,\z}}}]{x:=z}
\\
\finishblock[start ribbons={
  a/{left=0,right=10,label={\x\doteq 0}}
}]{\}}
\\
\extendboxes{alpha/{left=-8}}
\\[-9]
\extendboxes{beta/{left=-6}}
\\[-9]
\jus[finish ribbons={a,b},start ribbons={
  a/{right=\RIBbRIGHT,label=\alpha\doteq \varepsilon}}]
  {Unfold $\List$ def}
\swapribbons{c,d}{d,c}
\\
\jus[finish ribbons={a,d}, start ribbons={
  a/{right=\RIBdRIGHT,label=\delta\doteq\rev{\beta}}}]
  {Concatenate empty sequence}
\\
\jus[finish ribbons={a,c}, finish boxes={alpha,beta},
  start ribbons={
    a/{right=\RIBcRIGHT,label={\List\,\rev{\delta}\,\y}}}]
  {Fold $\List$ def}
\\
\end{ribbonproof}
```

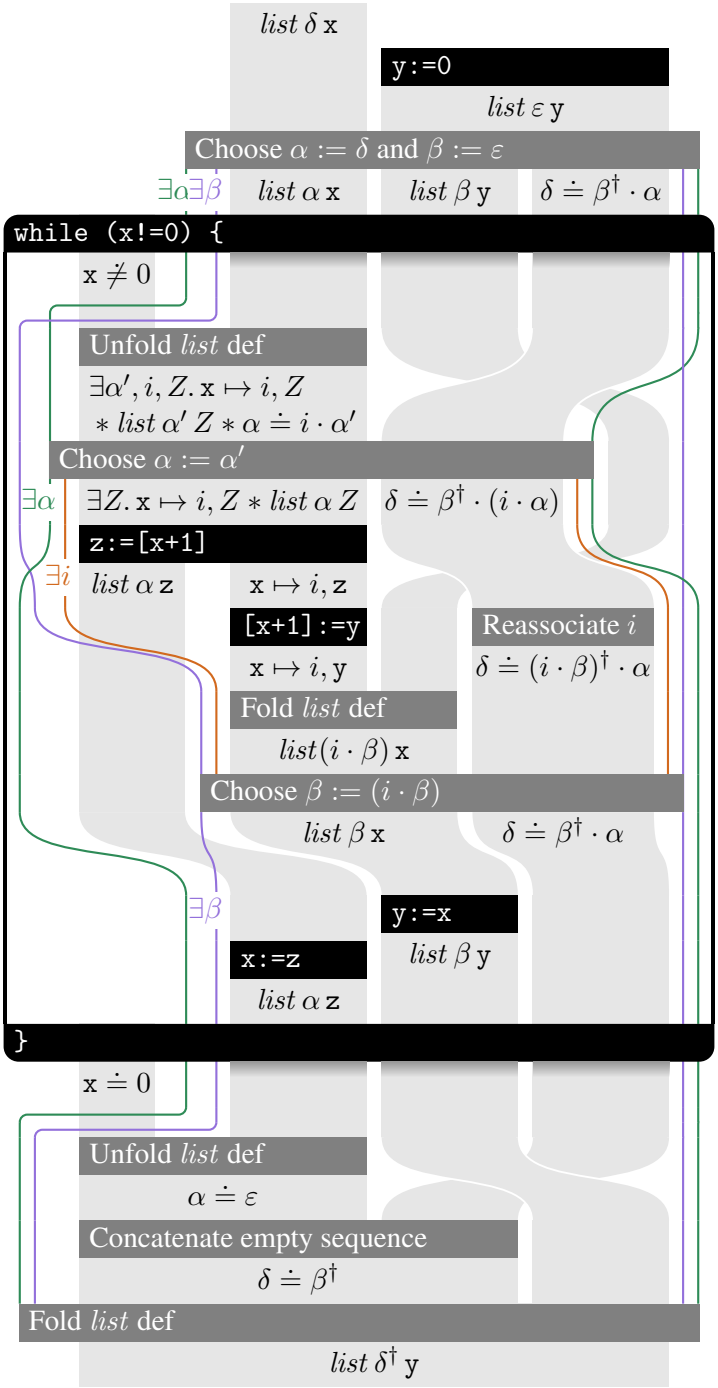


Figure 3: A ribbon proof of list reverse

2.3 List reverse with variables-as-resource

Figure 4 shows the result.

```
\begin{ribbonproof}[start ribbons={
  b/{left=20,right=38,var width=5,label=\x&\List\,\delta\,\x},
  e/{left=40,right=45, var width=5,label=\z&},
  c/{left=50,right=55, var width=5,label=\y&}}]
\jus[finish ribbons={b}, start ribbons={
  a/{left=-5,right=0,var width=5, label={\frac12\x&}},
  b/{label={\frac12\x&\List\,\delta\,\x}}
}]{Split $\x$}
\com[finish ribbons={c},start ribbons={
  c/{left=50,right=88,label=\y&\List\,\varepsilon\,\y}}]{y:=0};
\\
\jus[finish ribbons={b,c}, start ribbons={
  b/{label=\frac12\x&\List\,\alpha\,\x},
  c/{label=\y&\List\,\beta\,\y, right=68},
  d/{label=\delta\doteq \rev{\beta}\cdot\alpha, left=70, right=88}
}, start boxes={
  alpha/{left=14,right=94,label=\exists\alpha,color=SeaGreen},
  beta/{left=18,right=92,label=\exists\beta,color=MediumPurple}
}]{Choose $\alpha:=\delta$ and $\beta:=\varepsilon$}
\\
\startblock[fit boxes={alpha},
  extra left=10, extra right=2, finish ribbons={a}, start ribbons={
  a/{left=-5,right=10,label=\frac12\x&\x\dotneq 0}}]{while (x!=0) \{}
\\
\extendboxes{alpha/{left=-9}}
\\[-9]
\extendboxes{beta/{left=-13}}
\\[-9]
\jus[finish ribbons={a,b}, start ribbons={
  ab/{left=\RIBaLEFT, right=\RIBbRIGHT, var width=5,
  label={\x&\exists\alpha',i,Z\ldotp\x\mapsto i,Z \\{}
  * \List\,\alpha'\,Z * \alpha \doteq i \cdot \alpha'}}}]
  {Unfold $\List$ def}
\moveboxes{
  alpha/{right=92},
  beta/{right=94}}
\\[4]
\jus[finish boxes={alpha}, start boxes={
    alpha/{label=\exists\alpha},
    i/{left=-7,right=90,color=Chocolate,
    vertical offset=11,label=\exists i}
  }, finish ribbons={ab,d},
  start ribbons={
```

```
    ab/{label={\x&\exists Z\ldotp\x \mapsto i,Z * \List\,\alpha\,Z}},
    d/{label=\delta\doteq{}\\\rev{\beta}\cdot(i\cdot\alpha)}
  }]{Choose $\alpha:=\alpha'$}
\\
\com[finish ribbons={ab,e}, start ribbons={
  a/{right=14,label=\frac12\z&\List\,\alpha\,\z},
  b/{var width=10,left=16,right=\RIBeRIGHT,
    label={\x,\frac12\z&\x\mapsto i,\z}}
}]{z:=[x+1]}
\jus[finish ribbons={c}, start ribbons={
  cc/{left=57, right=\RIBcRIGHT, var width=5,
    rotated label={\List\,\beta\,\y}, label=\frac12\y&},
  c/{right=55, label=\frac12\y&}
}]{Split $\y$}
\moveboxes{alpha/{left=-11}}
\\
\com[finish ribbons={b,c}, start ribbons={
  b/{var width=5,right=21, label=\frac12\z&},
  c/{left=27, var width=10,label={\x,\frac12\y&\x\mapsto i,\y}}
}]{[x+1]:=y}
\jus[finish ribbons={d}, start ribbons={
  d/{label={\delta\doteq {}\\
   \rev{(i\cdot\beta)}\cdot\alpha}}}]{Reassoc. $i$}
\\[-6]
\moveboxes{
  i/{left=25},
  beta/{left=23,right=92},
  alpha/{right=94}}
\\
\jus[finish ribbons={a,b}, start ribbons={
  a/{right=\RIBbRIGHT, label={\z&\List\,\alpha\,\z}}
}]{Combine $\z$}
\jus[finish ribbons={c,cc}, start ribbons={
  c/{var width=5, label={\x&\List (i\cdot\beta)\,\x}},
  cc/{left=63, label=\y&}
}]{Fold $\List$ def}
\\
\jus[finish ribbons={c,d}, finish boxes={i,beta},
  start boxes={
    beta/{horizontal offset=2,label=\exists\beta}},
  start ribbons={
    c/{label={\List\,\beta\,\x}, label=\x&},
    d/{label={\delta\doteq\rev{\beta}\cdot\alpha}}}]
      {Choose $\beta:=(i\cdot\beta)$}
\\[-2]
\moveribbons{c/{left=43,right=61},a/{left=20,right=41}}
```

```
\moveboxes{alpha/{left=14},beta/{left=18}}
\\
\com[finish ribbons={c,cc}, start ribbons={
  e/{left=\RIBcLEFT, right=\RIBcLEFT+5, label=\x&},
  c/{left=50,right=\RIBccRIGHT,label={\List\,\beta\,\y}, label=\y&}
}]{y:=x}
\\
\com[finish ribbons={a,e}, start ribbons={
  b/{left=\RIBaLEFT,right=\RIBaRIGHT,label=\frac12\x&\List\,\alpha\,\x},
  a/{left=-5,right=0,var width=5, label=\frac12\x&},
  e/{label=\z&}
}]{x:=z}
\\
\finishblock[finish ribbons={a},start ribbons={
  a/{right=10,label={\frac12\x&\x\doteq 0}}
}]{\}}
\\
\extendboxes{alpha/{left=-8}}
\\[-9]
\extendboxes{beta/{left=-6}}
\\[-9]
\jus[finish ribbons={a,b},start ribbons={
  a/{right=0, label=\x&},
  b/{left=6,label=\alpha\doteq \varepsilon, var width=0}
}]{Unfold $\List$ def}
\moveribbons{
  d/{left=43,right=48},
  e/{left=74,right=79}
}
\moveboxes{
  alpha/{right=72},
  beta/{right=70}}
\\
\jus[finish ribbons={b,d}, start ribbons={
  b/{right=\RIBdRIGHT,label=\delta\doteq\rev{\beta}}
  }]{Concatenate empty seq.}
\moveboxes{alpha/{left=2},beta/{left=4}}
\\
\jus[finish ribbons={b,c}, finish boxes={alpha,beta},
  start ribbons={
    c/{label={\y&\List\,\rev{\delta}\,\y}}
  }]{Fold $\List$ def}
\\
\end{ribbonproof}
```

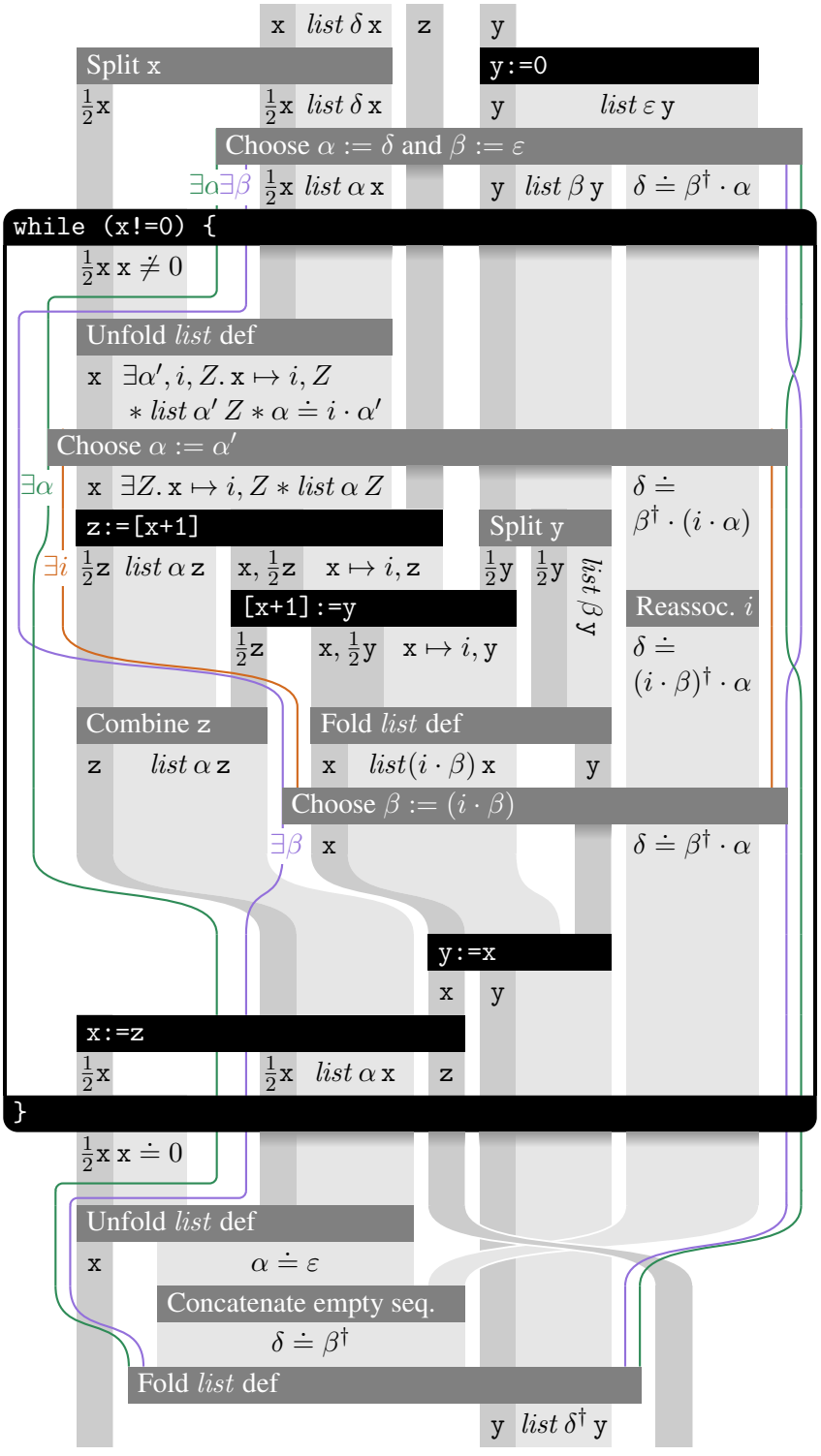


Figure 4: A ribbon proof of list reverse using variables-as-resource

3 Reference

3.1 Package dependencies

The ribbonproofs package relies upon:

- the xcolor package for colours;
- the tikz package, plus its calc, fadings and decorations.pathmorphing libraries, for drawing;
- the xstring package for manipulating strings; and
- the etextools package for controlling the expansion of TEX macros.

3.2 The `ribbonproof` environment

`\begin{ribbonproof}[`*options*`]`...`\end{ribbonproof}` This is the top-level environment for creating ribbon proofs. All of the ribbon-proof-building commands must only appear inside this environment. Available *options* are:

`extra height=`n (Default: `0`.) This option adds n (an integer or integer expression) to the height of the diagram's first row.

`start ribbons={`$ribbon_1$`,...,`$ribbon_k$`}` (Default: empty.) This defines a list of k ribbons that henceforth are *active*. Each $ribbon$ takes the form

$$name\texttt{/\{}options\texttt{\}}$$

where $name$ is an alphabetic string that henceforth identifies the ribbon, and the available *options* are:

`left=`n (Default: `0`.) The horizontal position of the ribbon's left edge. The value n must either be an integer or an expression that evaluates to an integer. The value is stored in the command `\RIBnameLEFT`, where `name` is the name of the ribbon. Ribbons can be positioned relatively by accessing this value. For instance, one can write `left=\RIBfooLEFT-20` to set the current ribbon's left edge 20 units to the left of the ribbon `foo`'s left edge.

`right=`n (Default: `0`.) The horizontal position of the ribbon's right edge. The value n must either be an integer or an expression that evaluates to an integer. The value is stored in the command `\RIBnameRIGHT`, where `name` is the name of the ribbon.

`label=`L (Default: empty.) The label that is written on the ribbon. The value L is parsed in math-mode. There are three complications here.

- If L contains a comma or an equals-sign, it must be wrapped in braces; otherwise the parser will become confused. For instance, be sure to write `label={0<x,y<5}` instead of `label=0<x,y<5`.
- *[Variables-as-resource only.]* If L contains an `&` symbol, the portion to the left of the `&` is used as the 'variable resource'. Here are some examples.

`x` $x < a$ `y` $\exists b.\, a = 2^b \wedge b < 9$

```
\begin{ribbonproof}[start ribbons={
  a/{left=5, right=25, var width=5,
     label=\texttt{x}&\texttt{x}<a},
  b/{left=30, right=35, var width=5,
     label=\texttt{y}&},
  c/{left=40, right=80,
     label={\exists b\ldotp a=2^b \wedge b<9}}}]
\end{ribbonproof}
```

`var width=`n (Default: `0`.) *[Variables-as-resource only.]* The width of the 'variable resource' part of the ribbon. The value is stored in the command `\RIBnameVARWIDTH`, where `name` is the name of the ribbon. In the case where

$$n = \texttt{right} - \texttt{left}$$

the ribbon consists *only* of the variable resource.

`rotated label=`L (Default: empty.) If this key is given instead of `label`, the label L will be rotated through 90° clockwise. This feature can be handy when horizontal space is tight.
[Variables-as-resource only.] It is not currently possible to rotate the variable resource. However, a variable resource can be coupled with a rotated assertion by using both the `label` and the `rotated label` keys in the following manner.


```
\begin{ribbonproof}[extra height=9, start ribbons={
  a/{left=0, right=11, var width=5,
    label=\texttt{x}&, rotated label=\texttt{x}<a}}]
\end{ribbonproof}
```

`start boxes={`$box_1, \ldots, box_k$`}` (Default: empty.) This defines a list of k boxes that henceforth are *active*. Each box takes the form

$$name\texttt{/\{}options\texttt{\}}$$

where $name$ is an alphabetic string that henceforth identifies the box, and the available $options$ are:

`left=`n (Default: `0`.) The horizontal position of the box's left edge. The value n must either be an integer or an expression that evaluates to an integer. The value is stored in the command `\BOXnameLEFT`, where `name` is the name of the box. Boxes can be positioned relatively by accessing this value. For instance, one can write `left=\BOXfooLEFT-5` to set the current box's left edge 5 units to the left of the box `foo`'s left edge.

`right=`n (Default: `0`.) The horizontal position of the box's right edge. The value n must either be an integer or an expression that evaluates to an integer. The value is stored in the command `\BOXnameRIGHT`, where `name` is the name of the box.

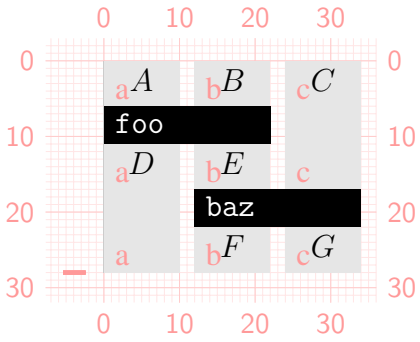
`label=`L (Default: empty.) The label that is written on the left edge of the box. The value L is parsed in math-mode.

`vertical offset=`n (Default: `0`.) Shifts the `label` down by n. The value n must either be an integer or an expression that evaluates to an integer.

`horizontal offset=`n (Default: `0`.) Shifts the `label` to the right by n. The value n must either be an integer or an expression that evaluates to an integer.

`color=`*c* (Default: `black`.) The colour of the box's lines and of its label text.

`draw grid` (Default: `false`.) This option lays a grid underneath the picture. The grid looks like graph paper, and is intended to help the user to position the elements of the diagram. As additional visual aids, it places little 'ticks' down the left-hand edge of the diagram, to show where each row begins, and it labels each ribbon with its name. Here is an example.




```
\begin{ribbonproof}[draw grid,
  start ribbons={
    a/{left=0, right=10, label=A},
    b/{left=12, right=22, label=B},
    c/{left=24, right=34, label=C}}]
\com[finish ribbons={a,b},
  start ribbons={
    a/{label=D},
    b/{label=E}}]{foo} \\
\com[finish ribbons={b,c},
  start ribbons={
    b/{label=F},
    c/{label=G}}]{baz} \\
\end{ribbonproof}
```

3.3 Commands

`\com[`*options*`]{`*text*`}` This creates a 'command step' in the proof; that is, a black rectangle, labelled with *text*, that has a precondition above it and a postcondition below it. Available *options* are:

`height=`*n* (Default: `5`.) The height of the step. The value *n* must either be an integer or an expression that evaluates to an integer. The default step height can be changed by renewing the `\defaultStepHeight` command.

`finish ribbons={`$name_1, \ldots, name_k$`}` (Default: empty.) The k ribbons that are identified by the given names are preconditions of this step. They are henceforth *inactive*.

`start ribbons` These ribbons are postconditions of this step. See page 14.

`finish boxes={`$name_1, \ldots, name_k$`}` (Default: empty.) The k boxes that are identified by the given names are preconditions of this step. They are henceforth *inactive*.

`start boxes` These boxes are postconditions of this step. See page 15.

`extra left=`*n* (Default: `0`.) Additional width on the lefthand side of the step. The value *n* must either be an integer or an expression that evaluates to an integer.

`extra right=`*n* (Default: `0`.) Additional width on the righthand side of the step. The value *n* must either be an integer or an expression that evaluates to an integer.

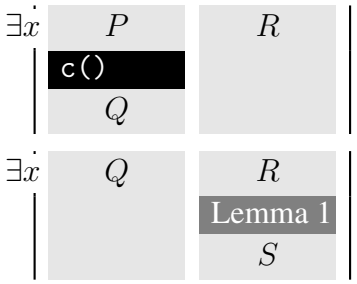
`\jus[`*options*`]{`*text*`}` This creates a 'justification step' in the proof; that is, a grey rectangle, labelled with *text*. The `\jus` command has exactly the same behaviour and options as the `\com` command, except:

- `\com` creates a black rectangle, but `\jus` creates a grey rectangle;

- `\com` sets its label in a monospaced font, but `\jus` sets its in a roman font; and
- `\com` sets its label on the left of the rectangle, but `\jus` sets its at the centre.

`\\[`*height*`]` Finish the current row and move to the next. The optional *height* parameter (a positive or negative integer, default `0`) is added to the default row height of `11`. This default row height can be changed by renewing the command `\defaultRowHeight`.

`\ribbonpagebreak` Insert a page break at this point in the proof. When the proof restarts on the next page, the labels on the ribbons and boxes are printed again, so that the reader does not have to turn back several pages to find the label of a long-lived ribbon or box. Here is an example. (The page break appears as a horizontal gap in the middle of the proof.)



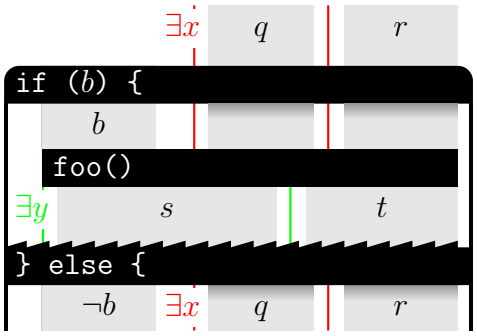

```
\begin{ribbonproof}[start ribbons={
  a/{left=0, right=18, label=P},
  b/{left=20, right=38, label=R}}, start boxes={
  x/{left=-2, right=40, label=\exists x}}]
\com[finish ribbons={a}, start ribbons={
  a/{label=Q}}]{c()} \\
\ribbonpagebreak
\jus[finish ribbons={b}, start ribbons={
  b/{label=S}}]{Lemma 1} \\
\end{ribbonproof}
```

`\startblock[`*options*`]{`*text*`}` This command begins a new *block*. A block is a compound step such as an if-statement or a while-loop. Blocks may contain further steps, or even further blocks (which must be well-nested). The first segment of a block resembles an ordinary command step – a black rectangle labelled with *text* – except that its top corners are rounded. (The radius of this rounding defaults to `2`, but this can be adjusted by renewing the `\roundingRadius` command.) Available *options* are `height`, `finish ribbons`, `start ribbons`, `finish boxes`, `start boxes`, `extra left` and `extra right`, which are the same as for the `\com` command: see page 16. Additional options are:

`fit ribbons={`$name_1,\ldots,name_k$`}` (Default: empty.) The k ribbons that are identified by the given names are used to calculate the width of this block. That is, the block must be wide enough to cover each of the named ribbons. Being one of the `fit ribbons` does not affect a ribbon in any way.

`fit boxes={`$name_1,\ldots,name_k$`}` (Default: empty.) As above, but for boxes instead of ribbons.

`\continueblock[`*options*`]{`*text*`}` This command continues a block. If the block represents an if-statement, then `\continueblock` corresponds to the 'else' keyword. Every ribbon or box inside the block is deactivated, and every ribbon or box that was inside the block at the beginning is restored. This segment of a block resembles an ordinary command step – a black rectangle labelled with *text* – except that its top edge is jagged (when the `jagged` key is provided) to suggest the discontinuity at this point in the proof. Here is an example.




```
\begin{ribbonproof}[extra height=2,
  start ribbons={
    q/{left=12, right=26, label=q},
    r/{left=30, right=45, label=r}},
  start boxes={
    x/{left=10, right=28, color=red,
      label=\exists x}}]
\startblock[fit ribbons={r},
  extra left=5, extra right=2,
  start ribbons={p/{left=-10, right=5,
      label=b}}]{if ($b$) \{} \\
\com[finish ribbons={p,q,r},
  finish boxes={x}, start ribbons={
    s/{left=-8, right=21, label=s},
    t/{left=25, right=45, label=t}},
  start boxes={
    y/{left=-10, right=23, color=green,
      label=\exists y}}]{foo()} \\[2]
\continueblock[jagged, repeat labels,
  start ribbons={
    p/{label=\neg b}}]{\} else \{} \\
\end{ribbonproof}
```

Available *options* are `height`, `start ribbons` and `start boxes`, which are the same as for the `\com` command: see page 16. Additional options are:

`jagged` (Default: none.) If this option is present, the top edge of the rectangle is drawn jagged. If not, the top edge is simply a straight line.

`repeat labels` (Default: none.) If this option is present, each ribbon or box that is restored by this step is given a new label, as a reminder to the reader. If not, labels are only printed for those ribbons and boxes given explicitly in the `start ribbons` or `start boxes` lists.

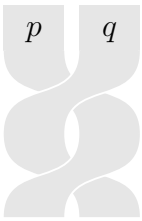
`\finishblock[`*options*`]{`*text*`}` This command ends a block. If the block represents an if-statement or a while-loop, then `\finishblock` corresponds to the 'end-if' or 'end-while' keyword. Unlike the `\continueblock` command, `\finishblock` does not deactivate or restore any ribbons or boxes – except for those explicitly given as `start ribbons`, `start boxes`, `finish ribbons` or `finish boxes`. This final segment of a block resembles an ordinary command step – a black rectangle labelled with *text* – except that its bottom corners are rounded. (The radius of this rounding defaults to 2, but this can be adjusted by renewing the `\roundingRadius` command.) Available *options* are `height`, `finish ribbons`, `start ribbons`, `finish boxes`, `start boxes`, `extra left` and `extra right`, which are the same as for the `\com` command: see page 16.

`\moveribbons{`$ribbon_1, \ldots, ribbon_k$`}` This command provides an opportunity to redefine the `left` and `right` positions of some ribbons. Each *ribbon* takes the form

$$name/\{options\}$$

where *name* identifies an active ribbon, and the available *options* are `left`, `right` and `var width`, which are the same as for the `start ribbons` key: see page 14. At the end

of the current row, the left and right positions of each ribbon are examined, and if they have changed as the result of a \moveribbons command, a smooth shape is drawn from the old positions to the new positions. A thin white border is added to the left and right edges of each ribbon, to make overlapping ribbons easier to distinguish. The z-order of ribbons can be controlled by their position in the list provided to \moveribbons: later ribbons are drawn over earlier ones. Here is an example that illustrates how to control overlapping in this way.




```
\begin{ribbonproof}[start ribbons={
  p/{left=0, right=8, label=p},
  q/{left=10, right=18, label=q}}]
\moveribbons{p/{left=10, right=18}, q/{left=0, right=8}} \\
\moveribbons{q/{left=10, right=18}, p/{left=0, right=8}} \\
\end{ribbonproof}
```

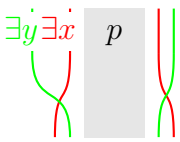
\swapribbons{$name_1,\ldots,name_k$}{$name'_1,\ldots,name'_k$} This command provides an easy way to permute a number of ribbons. Each $name$ identifies an active ribbon. The second argument is expected to contain the same names as the first, but perhaps in a different order. The effect is that for each i simultaneously, the new position of the ribbon identified by $name'_i$ becomes the old position of the ribbon identified by $name_i$. The z-order of ribbons can be controlled by their position in the second argument: later ribbons are drawn over earlier ones. Here is an example that illustrates how to control overlapping in this way.


```
\begin{ribbonproof}[start ribbons={
  p/{left=0, right=8, label=p},
  q/{left=10, right=18, label=q}}]
\swapribbons{q,p}{p,q} \\
\swapribbons{p,q}{q,p} \\
\end{ribbonproof}
```

\moveboxes{$box_1,\ldots,box_k$} This command provides an opportunity to redefine the left and right positions of some boxes. Each box takes the form

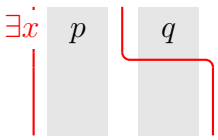
$$name/\{options\}$$

where $name$ identifies an active box, and the available $options$ are left and right, which are the same as for the start boxes key: see page 15. At the end of the current row, the left and right positions of each box are examined, and if they have changed as the result of a \moveboxes command, a smooth shape is drawn from the old positions to the new positions. Here is an example.




```
\begin{ribbonproof}[start ribbons={
    p/{left=7, right=15, label=p}},
  start boxes={
    x/{left=5, right=17, color=red, label=\exists x},
    y/{left=0, right=19, color=green, label=\exists y}}]
\moveboxes{x/{left=3, right=19}, y/{left=5, right=17}} \\
\end{ribbonproof}
```

`\extendboxes{`$box_1, \ldots, box_k$`}` This command accepts the same input as the `\moveboxes` command. Like the `\moveboxes` command, it allows the `left` and `right` positions of some boxes to be modified. However, where `\moveboxes` draws smooth curves between the old and new positions, `\extendboxes` draws a 'rounded elbow' shape between them. Such a shape may be preferable when extending a box's scope to include additional ribbons. Here is an example.




```
\begin{ribbonproof}[start ribbons={
    p/{left=8, right=16, label=p},
    q/{left=20, right=28, label=q}},
  start boxes={
    x/{left=6, right=18, color=red, label=\exists x}}]
\extendboxes{x/{right=30}} \\
\end{ribbonproof}
```

3.4 Adjustable parameters

Each of the following parameters can be adjusted using `\renewcommand`.

`\jusColor` (Default: `black!50`). Background colour of 'justification' steps.

`\comColor` (Default: `black`). Background colour of 'command' steps.

`\ribColor` (Default: `black!10`). Background colour of ribbons.

`\varribColor` (Default: `black!20`). *[Variables-as-resource only.]* Background colour of the variable resource part of a ribbon.

`\ribTextColor` (Default: `black`). Colour of the textual label on ribbons.

`\boxTextColor` (Default: `black`). Colour of the textual label on boxes.

`\guideTextColor` (Default: `red!40`). Colour of the 'guide' text (that is, the ribbon names and numeric scale that are printed when `draw grid` is enabled).

`\defaultStepHeight` (Default: `5`). Default height of a step.

`\defaultRowHeight` (Default: `11`). Default height of a row. (This includes the step.)

`\ribTextVOffset` (Default: `4`). Vertical adjustment of ribbon labels.

`\boxTextVOffset` (Default: `3`). Vertical adjustment of box labels.

`\boxTextHOffset` (Default: `1`). Horizontal adjustment of box labels.

`\guideTextVOffset` (Default: `2`). Vertical adjustment of guide text on ribbons.

`\roundingRadius` (Default: `2`). Radius of rounded corners on blocks.

`\boxRoundingRadius` (Default: `1`). Radius of rounded corners on boxes. (Applies to the `\extendboxes` command.)

`\blockLineWidth` (Default: `0.5`). Width of the vertical lines that delimit blocks.

`\boxLineWidth` (Default: `0.8pt`). Width of the vertical lines that delimit boxes.

`\shadowHeight` (Default: `2`). Height of the shadows that are drawn when a step passes over a ribbon without affecting it.

`\shadowColor` (Default: `black!40`). Colour of the darkest point of a shadow.

`\zigzagHeight` (Default: `1`). Height of the jagged edges that are drawn on the top edge of each `\continueblock` step.

`\zigzagLength` (Default: `3`). Length of the jagged edges that are drawn on the top edge of each `\continueblock` step.

`\twistiness` (Default: `0.7`). For adjusting the shape of the ribbon twists. Must be a value between 0 and 1 inclusive.

Bibliography

[1] J. Bean. *Ribbon Proofs - A Proof System for the Logic of Bunched Implications*. PhD thesis, Queen Mary University of London, 2006.

[2] P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.

[3] J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In G. D. Plotkin, editor, *Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS '02)*. IEEE Computer Society, 2002.

[4] J. Wickerson, M. Dodds, and M. J. Parkinson. Ribbon proofs for separation logic. In M. Felleisen and P. Gardner, editors, *Proceedings of the 22nd European Symposium on Programming (ESOP '13)*, volume 7792 of *Lecture Notes in Computer Science*, pages 189–208. Springer, 2013.

Index