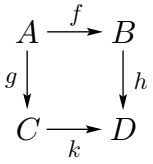
NCSq Examples

Taichi Uemura

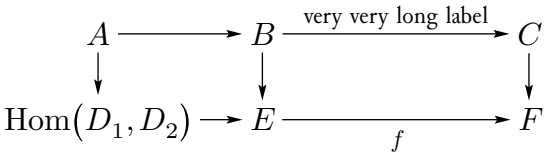
To draw a diagram, pass `\NCSq.diagram` a function building a rectangular array of objects, horizontal arrows and vertical arrows. (Diagonal arrows are not supported.)

```
\NCSq.diagram
(fun tb ->
 (let emp = tb#emp in
  let obj = tb#obj in
  let harr = tb#harr in
  let varr = tb#varr in
  [[obj {${A}}    ; harr {${f}} {}; obj {${B}}    ];
   [varr {} {${g}}; emp           ; varr {${h}} {}];
   [obj {${C}}    ; harr {} {${k}}; obj {${D}}    ]]));
```



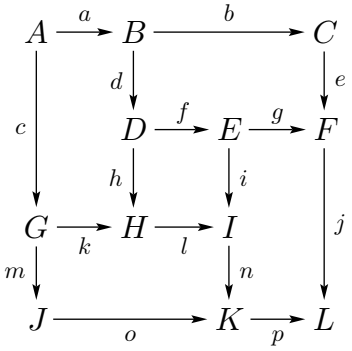
Arrows stretch automatically.

```
\NCSq.diagram
(fun tb ->
 (let emp = tb#emp in
  let obj = tb#obj in
  let harr = tb#harr in
  let varr = tb#varr in
  [[obj {${A}}; harr {} {}; obj {${B}};
    harr {very very long label} {}; obj {${C}}];
   [varr {} {}; emp; varr {} {}; emp; varr {} {}];
   [obj {${\mathrm{Hom}\paren{D_{1}, D_{2}}}}; harr {} {};
    obj {${E}}; harr {} {${f}}; obj {${F}}]]));
```



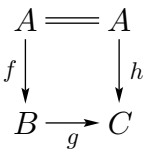
The source and the target of an arrow are automatically detected: the source of a horizontal arrow is the nearest object on the left; the target of a horizontal arrow is the nearest object on the right; the source of a vertical arrow is the nearest object above; the target of a vertical arrow is the nearest object below. (If the source or the target of an arrow is not found, then the arrow will not be drawn.)

```
\NCSq.diagram(fun tb -> (
  let emp = tb#emp in
  let obj = tb#obj in
  let harr = tb#harr in
  let varr = tb#varr in
  [[obj {${A}}; harr {${a}} {}; obj {${B}}; harr {${b}} {};
    emp; emp; obj {${C}}];
   [varr {} {${c}}; emp; varr {} {${d}}; emp;
    emp; emp; varr {${e}} {}];
   [emp; emp; obj {${D}}; harr {${f}} {};
    obj {${E}}; harr {${g}} {}; obj {${F}}];
   [emp; emp; varr {} {${h}}; emp;
    varr {${i}} {}; emp; varr {${j}} {}];
   [obj {${G}}; harr {} {${k}}; obj {${H}}; harr {} {${l}};
    obj {${I}}];
   [varr {} {${m}}; emp; emp; emp;
    varr {${n}} {}];
   [obj {${J}}; harr {} {${o}}; emp; emp;
    obj {${K}}; harr {} {${p}}; obj {${L}}]]));
```



heq and veq draw horizontal and vertical, respectively, equality signs, so we can now draw a commutative "triangle".

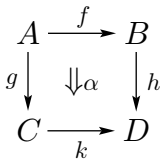
```
\NCSq.diagram(fun tb -> (
  let emp = tb#emp in
  let obj = tb#obj in
  let harr = tb#harr in
  let varr = tb#varr in
  let heq = tb#heq in
  [[obj {${A}}; heq; obj {${A}}];
   [varr {} {${f}}; emp; varr {${h}} {}];
   [obj {${B}}; harr {} {${g}}; obj {${C}}]]));
```



One can also draw 2-arrows. (Internally, a 2-arrow is an object to which "⇓" is appended.)

```
\NCSq.diagram(fun tb -> (
  let obj = tb#obj in
  let harr = tb#harr in
  let varr = tb#varr in
  let arr2 = tb#arr2 in
  [[obj {${A}}; harr {${f}} {}; obj {${B}}];
   [varr {} {${g}}; arr2 {${\alpha}}; varr {${h}} {}];
```

```
   [obj {${C}}; harr {} {${k}}; obj {${D}}]]));
```



`\NCSq.diagram-m` is suitable for drawing diagrams in math formulas.

```
\eqn(${
  \paren{
    \NCSq.diagram-m
    !(fun tb -> (
      let obj = tb#obj in
      let harr = tb#harr in
      let varr = tb#varr in
      let heq = tb#heq in
      let veq = tb#veq in
      let arr2 = tb#arr2 in
      [[obj {${A}}; harr {${f}} {}; obj {${B}}; heq; obj {${B}}];
       [veq; arr2 {${\epsilon}};
        varr {${g}} {}; arr2 {${\eta}}; veq];
       [obj {${A}}; heq; obj {${A}}; harr {} {${f}}; obj {${B}}]]
    ))
  } =
  \paren{
    \NCSq.diagram-m
    !(fun tb -> (
      let emp = tb#emp in
      let obj = tb#obj in
      let harr = tb#harr in
      let veq = tb#veq in
      [[obj {${A}}; harr {${f}} {}; obj {${B}}];
       [veq; emp; veq];
```

```
       [obj {${A}}; harr {} {${f}}; obj {${B}}]]))
  }
});
```

$$\left(\begin{array}{ccc} A & \xrightarrow{f} & B \\ \| & \Downarrow\varepsilon & \downarrow g \\ A & = & A \end{array} \begin{array}{ccc} = & B & \\ \Downarrow\eta & \| & \\ \xrightarrow[f]{} & B & \end{array}\right) = \left(\begin{array}{ccc} A & \xrightarrow{f} & B \\ \| & & \| \\ A & \xrightarrow[f]{} & B \end{array}\right)$$

Arrows can be backwards.

```
\NCSq.diagram
(fun tb -> (
  let obj = tb#obj in
  let rarr = tb#rarr in
  let uarr = tb#uarr in
  let darr = tb#darr in
  let arr2 = tb#arr2 in
  [[obj {${C}}; rarr {よ} {};
    obj {${\app{\mathbf{Fun}}{C^{\mathrm{op}}, \mathbf{Set}}}}];
   [darr {} {${F}}; arr2 {}; uarr {} {${F^{*}}}];
   [obj {${D}}; rarr {} {よ};
    obj {${\app{\mathbf{Fun}}{D^{\mathrm{op}}, \mathbf{Set}}}}]]));
```

$$\begin{array}{ccc} C & \xrightarrow{よ} & \mathbf{Fun}\big(C^{\mathrm{op}}, \mathbf{Set}\big) \\ {\scriptstyle F}\downarrow & \Downarrow & \uparrow{\scriptstyle F^{*}} \\ D & \xrightarrow[よ]{} & \mathbf{Fun}\big(D^{\mathrm{op}}, \mathbf{Set}\big) \end{array}$$

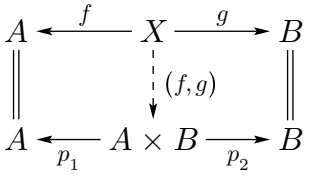
Here,

- `larr` = left arrow
- `rarr` = right arrow
- `uarr` = up arrow
- `darr` = down arrow

In fact, `harr` and `varr` are aliases to `rarr` and `darr`, respectively.

Arrow functions such as `rarr` and `darr` accept an optional argument to change the style.

```
\NCSq.diagram
(fun tb -> (
  let emp = tb#emp in
  let obj = tb#obj in
  let larr = tb#larr in
  let rarr = tb#rarr in
  let darr = tb#darr in
  let veq = tb#veq in
  [[obj {${A}}; larr {} {${f}}; obj {${X}};
    rarr {${g}} {}; obj {${B}}];
   [veq; emp; darr ?:(Dashed) {${\paren{f, g}}} {}; emp; veq];
   [obj {${A}}; larr {${p_{1}}} {}; obj {${A \times B}};
    rarr {} {${p_{2}}}; obj {${B}}]]));
```



Currently, two styles `Solid` (default) and `Dashed` are supported.