Harmony

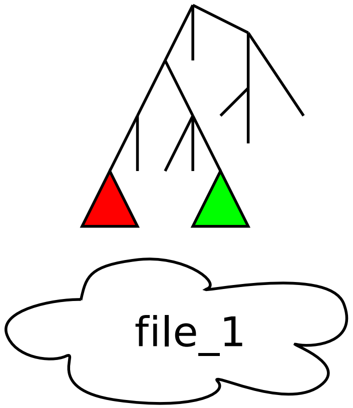
A Framework for Heterogeneous Data Synchronization

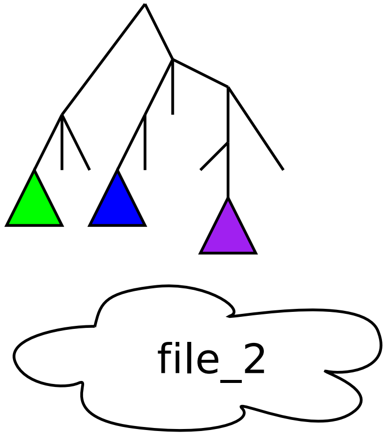
Michael Greenwald, Owen Gunden, Jon Moore,
Benjamin Pierce, Alan Schmitt, Stephen Tse

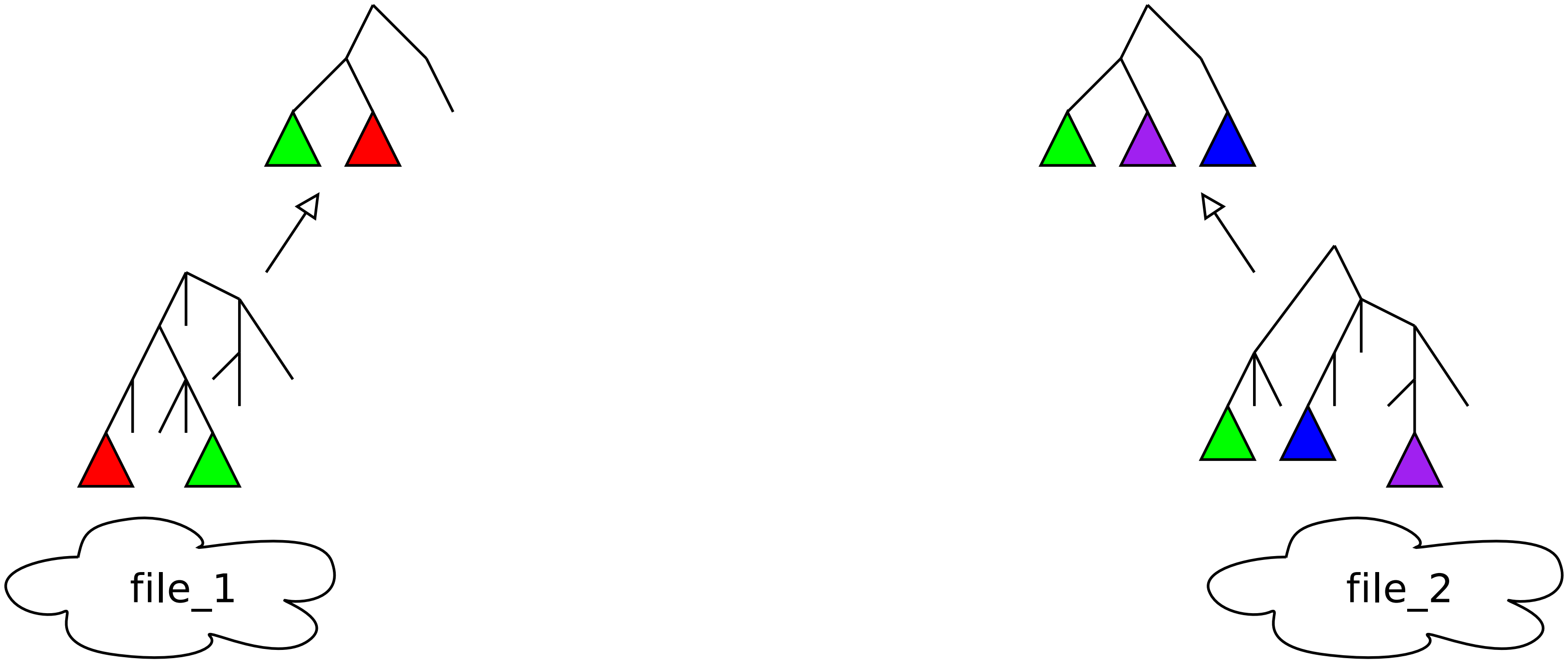
University of Pennsylvania

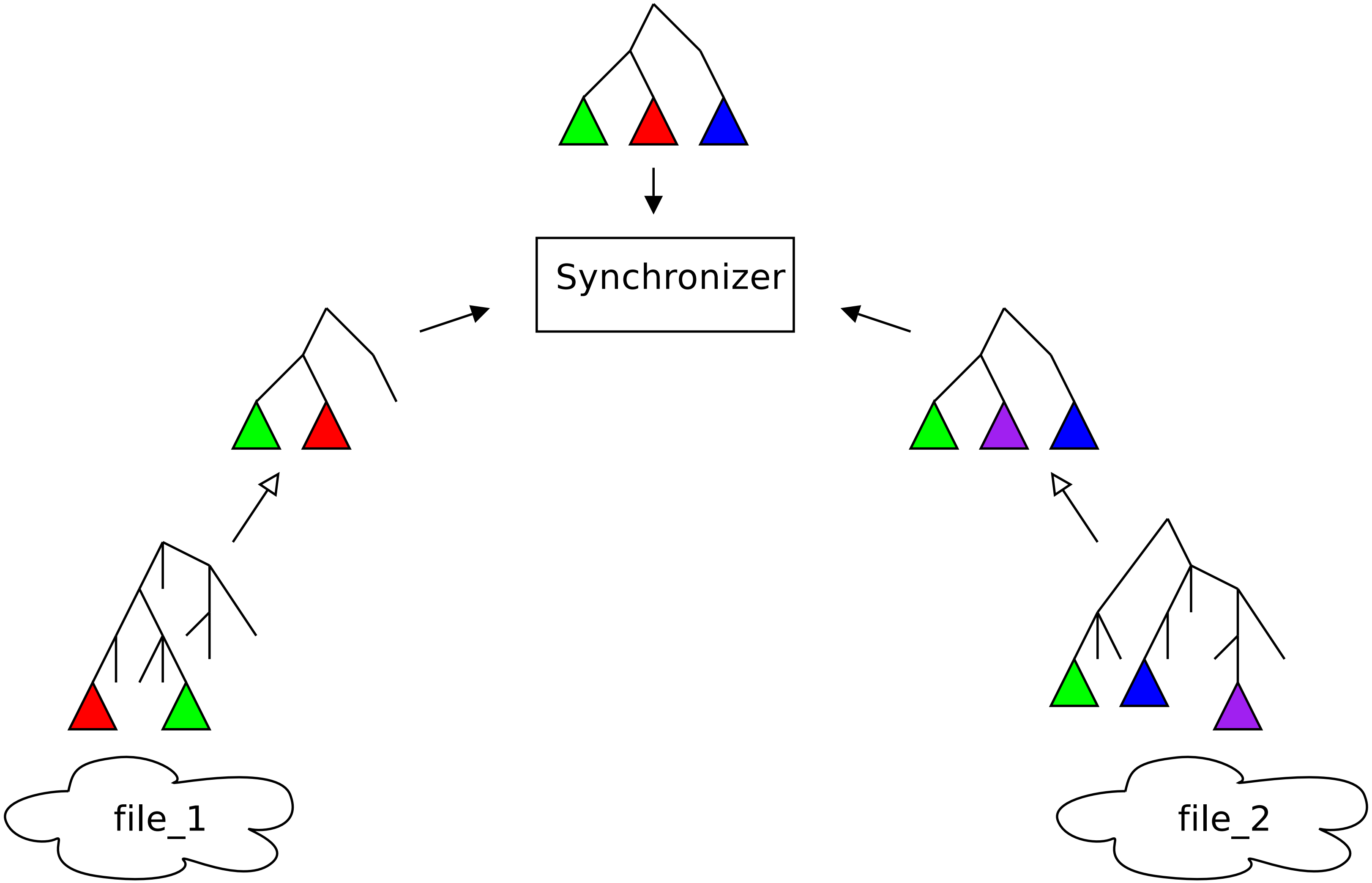
“Building the next 700 synchronizers...”

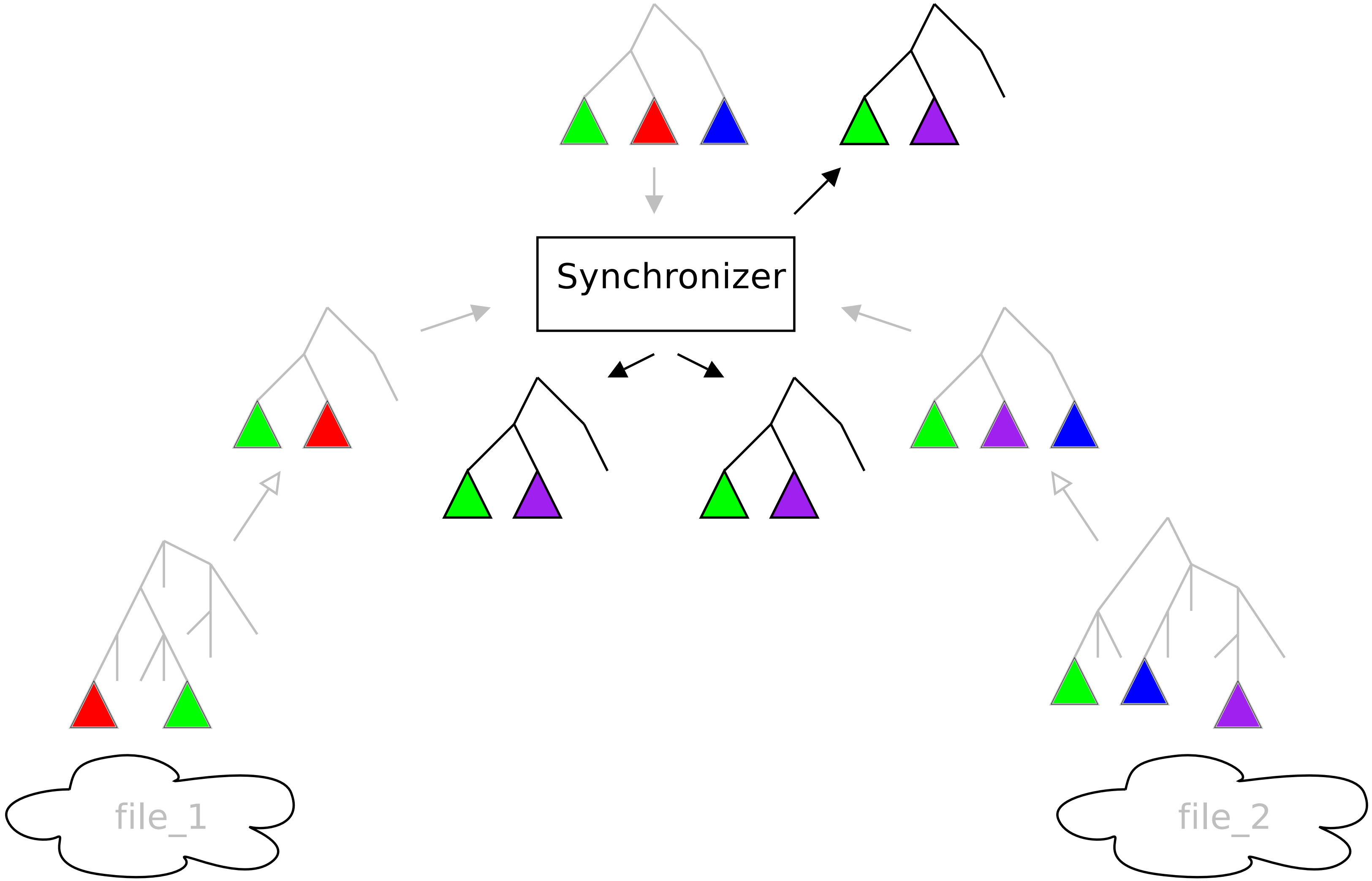
file_1
file_2

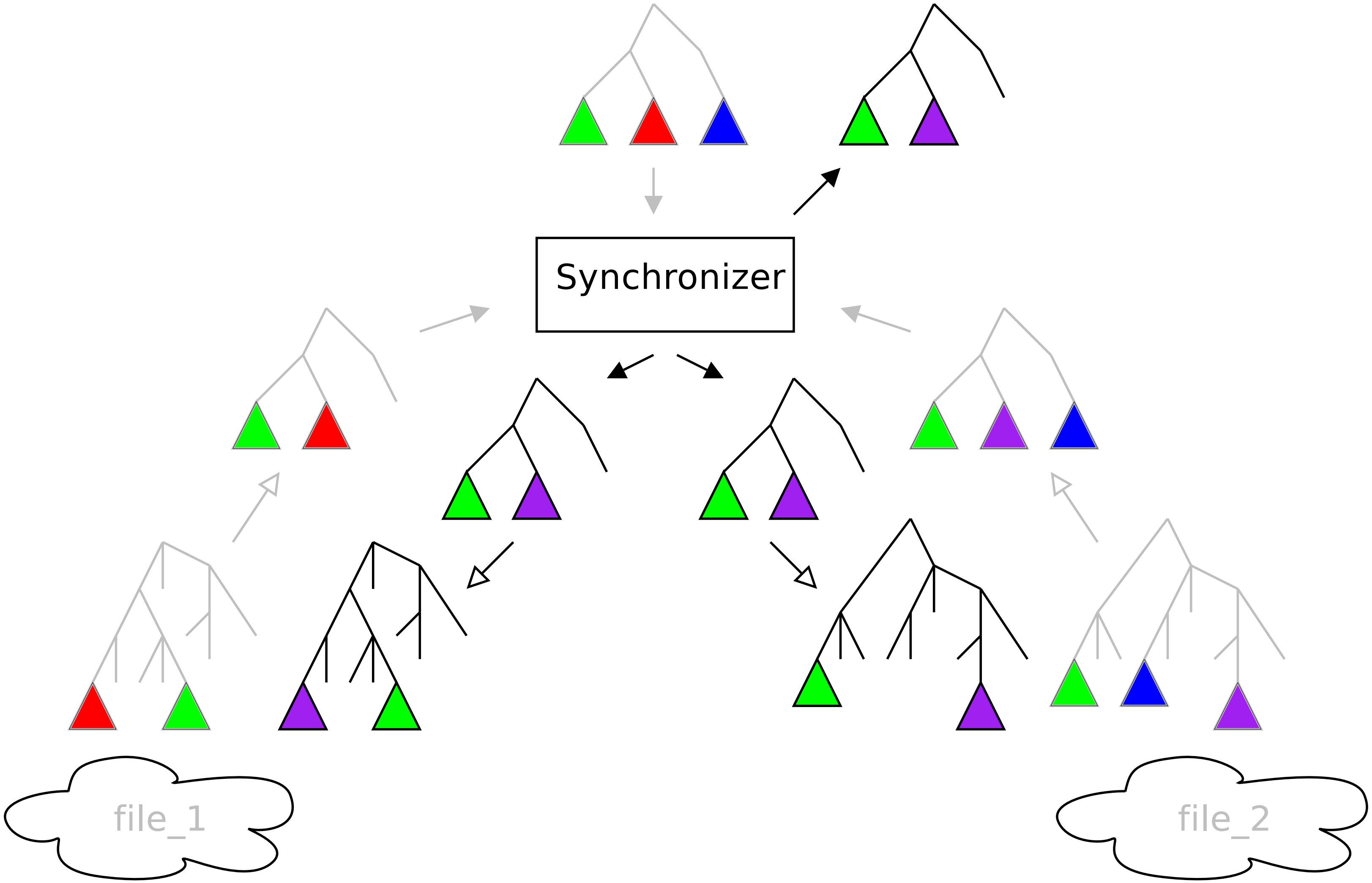

file_1

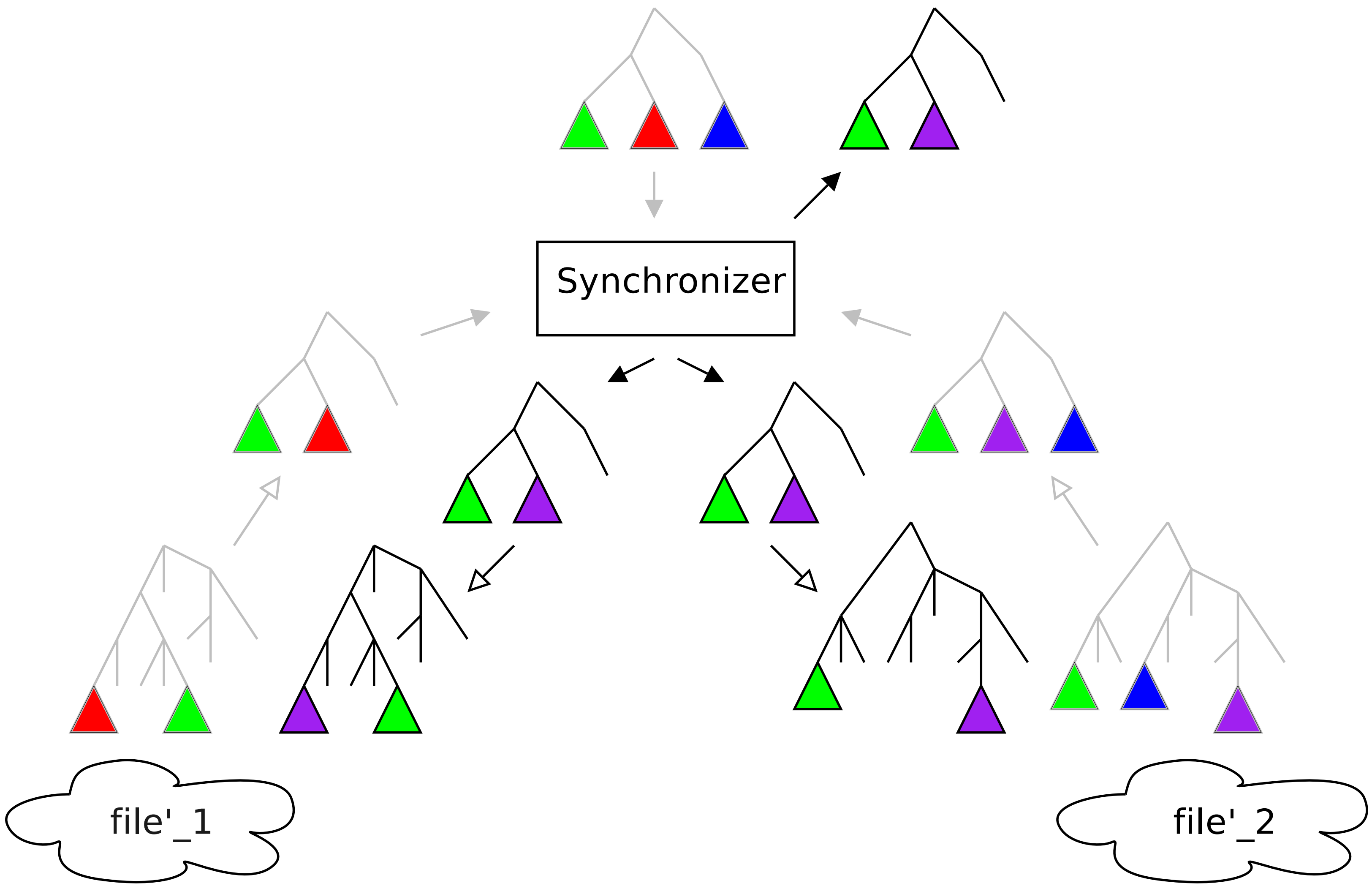

file_2


file_1
file_2


Synchronizer
file_1
file_2


Synchronizer
file_1
file_2


Synchronizer
file_1
file_2


Synchronizer
file'_1
file'_2

Views

Unordered, edge-labeled trees. Formally:

$$V \quad = \quad name \longrightarrow V$$

$$c \quad = \quad \left\{ \begin{array}{l} \texttt{Pat} \mapsto \left\{ \begin{array}{l} \texttt{Phone} \mapsto \texttt{333-4444} \\ \texttt{URL} \mapsto \texttt{http://pat.com} \end{array} \right. \\ \texttt{Chris} \mapsto \left\{ \begin{array}{l} \texttt{Phone} \mapsto \texttt{888-9999} \\ \texttt{URL} \mapsto \texttt{http://chris.org} \end{array} \right. \end{array} \right.$$

List encoding

```
[v1 v2 ... vn]
```

$$\left\{\begin{array}{l} \texttt{*h} \mapsto \texttt{v}_1 \\ \texttt{*t} \mapsto \left\{\begin{array}{l} \texttt{*h} \mapsto \texttt{v}_2 \\ \texttt{*t} \mapsto \left\{ \ldots \mapsto \left\{\begin{array}{l} \texttt{*h} \mapsto \texttt{v}_n \\ \texttt{*t} \mapsto \{ \end{array}\right.\right.\end{array}\right.\end{array}\right.$$

XML encoding

```
<tag attr1="val1" ... attrm="valm">
   subelt1 ... subeltn
</tag>
```

$$\left\{ \texttt{tag} \mapsto \left\{ \begin{array}{l} \texttt{attr1} \mapsto \texttt{val1} \\ \quad\vdots \\ \texttt{attrm} \mapsto \texttt{valm} \\ \texttt{''} \mapsto \left[\begin{array}{c} \langle\texttt{subelt1}\rangle \\ \vdots \\ \langle\texttt{subeltn}\rangle \end{array} \right. \end{array} \right. \right.$$

Lenses, Formally

- Set A of *abstract views*
- Set C of *concrete views*

Pair of partial functions

- A *get* function $l\nearrow$ from C to A
- A *put* function $l\searrow$ from $A \times C$ to C

Lens laws

- GetPut: $l\searrow(l\nearrow c, c) = c$ (the *put* function has no side effect)
- PutGet: $l\nearrow l\searrow(a, c) = a$ (the *put* function propagates all data)

Connection to Database theory

Strongly connected to the *view update problem*: given an update on some abstract view, find a translation of this update on the concrete view

- many such updates may exist
- notion of *complement* of a view:
  - a view and its complement include all information from the concrete view
- update under *constant complement*:
  - the translation of an update does not modify the complement

Some lenses

$$\boxed{\begin{array}{rcl}
\texttt{id} \nearrow c & = & c \\
\texttt{id} \searrow (a,\, c) & = & a
\end{array}}$$

$$\boxed{\begin{array}{rcll}
(\texttt{const}\ v\ d) \nearrow c & = & v & \\
(\texttt{const}\ v\ d) \searrow (a,\, c) & = & c & \text{if } a = v \text{ and } c \neq \Omega \\
 & & d & \text{if } a = v \text{ and } c = \Omega \\
 & & \text{undef.} & \text{otherwise}
\end{array}}$$

$$\boxed{\begin{array}{rcll}
(l; k) \nearrow c & = & k \nearrow l \nearrow c & \\
(l; k) \searrow (a,\, c) & = & l \searrow (k \searrow (a,\, l \nearrow c),\, c) & \text{if } c \neq \Omega \\
 & & l \searrow (k \searrow (a,\, \Omega),\, \Omega) & \text{if } c = \Omega
\end{array}}$$

More lenses

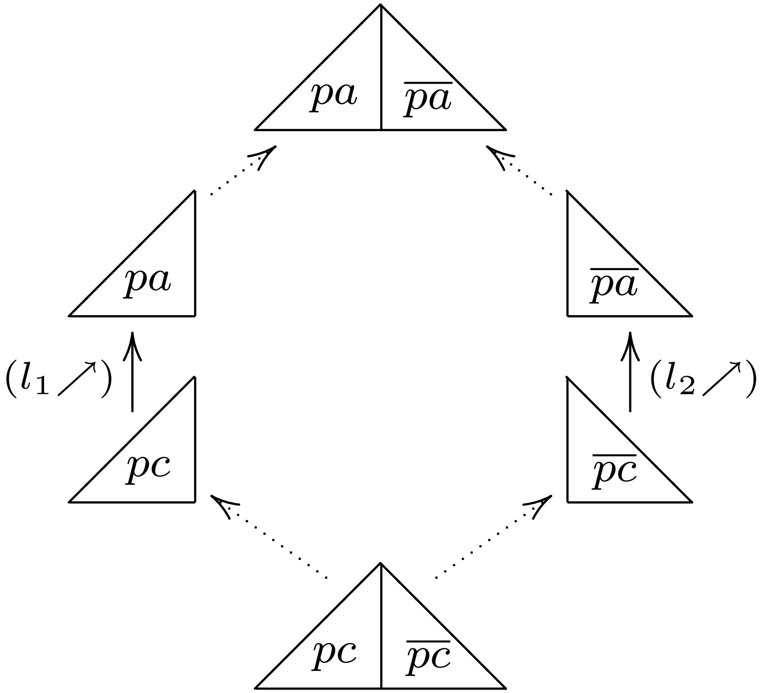
$$\boxed{\begin{aligned}(\texttt{rename}\ b)\nearrow c &= \left\{b(n) \mapsto c(n)\right.\\ (\texttt{rename}\ b)\searrow (a,\, c) &= \left\{b^{-1}(n) \mapsto a(n)\right.\end{aligned}}$$

$$\boxed{\begin{array}{rcll}(\texttt{hoist}\ n)\nearrow c &=& v & \text{if } c = \left\{n \mapsto v\right. \\ && \text{undef.} & \text{otherwise} \\ (\texttt{hoist}\ n)\searrow (a,\, c) &=& \left\{n \mapsto a\right. & \end{array}}$$

More lenses (2)

$$\begin{array}{rcll} (\mathtt{pivot}\ n)\nearrow c & = & \left\{k \mapsto v\right. & \text{if } c = \left\{\begin{array}{l} n \mapsto \left\{k \mapsto \{\right. \\ v \end{array}\right. \\ & & \text{undef.} & \text{otherwise} \\ (\mathtt{pivot}\ n)\searrow(a, c) & = & \left\{\begin{array}{l} n \mapsto \left\{k \mapsto \{\right. \\ v \end{array}\right. & \text{if } a = \left\{k \mapsto v\right. \\ & & \text{undef.} & \text{otherwise} \end{array}$$

Lens combinators: `xfork`



Lens combinators: `xfork`

$$
\begin{array}{lll}
(\texttt{xfork}\ pc\ pa\ l_1\ l_2) \nearrow c & = & \\
 & (l_1 \nearrow c|_{pc}) + (l_2 \nearrow c|_{\overline{pc}}) & \text{if (1)} \\
 & \text{undef.} & \text{otherwise} \\
(\texttt{xfork}\ pc\ pa\ l_1\ l_2) \searrow (a,\ c) & = & \\
 & (l_1 \searrow (a|_{pa},\ c|_{pc})) + (l_2 \searrow (a|_{\overline{pa}},\ c|_{\overline{pc}})) & \text{if (2)} \\
 & \text{undef.} & \text{otherwise}
\end{array}
$$

$$
\begin{array}{lll}
(1) & & \text{dom}(l_1 \nearrow c|_{pc}) \subseteq pa \\
 & \wedge & \text{dom}(l_2 \nearrow c|_{\overline{pc}}) \subseteq \overline{pa} \\
(2) & & \text{dom}(l_1 \searrow (a|_{pa},\ c|_{pc})) \subseteq pc \\
 & \wedge & \text{dom}(l_2 \searrow (a|_{\overline{pa}},\ c|_{\overline{pc}})) \subseteq \overline{pc}
\end{array}
$$

Lens combinators: `map`

$$(\texttt{map}\ l) \nearrow c \quad = \quad \left\{ n \mapsto l \nearrow c(n) \quad n \in \mathsf{dom}(c) \right.$$

$$(\texttt{map}\ l) \searrow (a,\, c) \quad = \quad \begin{cases} n \mapsto l \searrow (a(n),\, c(n)) & \\ \qquad\qquad n \in \mathsf{dom}(a) \cap \mathsf{dom}(c) & \\ n \mapsto l \searrow (a(n),\, \Omega) & \\ \qquad\qquad n \in \mathsf{dom}(a) \setminus \mathsf{dom}(c) & \end{cases}$$

Derived lenses

$$\begin{aligned}
\mathtt{fork}\ p\ l_1\ l_2 &= \mathtt{xfork}\ p\ p\ l_1\ l_2 \\
\mathtt{filter}\ p\ d &= \mathtt{fork}\ p\ \mathtt{id}\ (\mathtt{const}\ \{\}\ d) \\
\mathtt{prune}\ n\ d &= \mathtt{fork}\ \overline{\{n\}}\ \mathtt{id}\ \ (\mathtt{const}\ \{\}\ \{n \mapsto d\}) \\
\mathtt{focus}\ n\ d &= (\mathtt{filter}\ \{n\}\ d);\ (\mathtt{hoist}\ n) \\
\mathtt{mapp}\ p\ l &= \mathtt{fork}\ p\ (\mathtt{map}\ l)\ \mathtt{id}
\end{aligned}$$

$$\begin{aligned}
\mathtt{dispatch}\ [\,] &= \mathtt{id} \\
\mathtt{dispatch}\ (pc, pa, l) :: rest &= \mathtt{xfork}\ pc\ pa\ l\ \ (\mathtt{dispatch}\ rest)
\end{aligned}$$

Derived lenses (for lists)

$$\texttt{hd}\ d = \texttt{focus}\ \{\texttt{*h}\}\ \{\texttt{*t} \mapsto d\}$$

$$\texttt{tl}\ d = \texttt{focus}\ \{\texttt{*t}\}\ \{\texttt{*h} \mapsto d\}$$

$$\texttt{map\_list}\ l = \texttt{mapp}\ \{\texttt{*h}\}\ l;\ \texttt{mapp}\ \{\texttt{*t}\}\ (\texttt{map\_list}\ l)$$

$$\begin{aligned}
\texttt{hoist\_list}\ [\,] &= \texttt{id} \\
\texttt{hoist\_list}\ p :: \mathit{rest} &= \texttt{xfork}\ \{\texttt{*h}\}\ p \\
&\quad (\texttt{hoist}\ \{\texttt{*h}\}) \\
&\quad (\texttt{hoist}\ \{\texttt{*t}\};\ \texttt{hoist\_list}\ \mathit{rest})
\end{aligned}$$

Applications

- generic synchronizers (XML, HTML, meta, outline . . .)
- (mostly finished): “universal bookmark synchronizer”
- (in progress): multi-format calendar syncing (Palm, ical, iCalendar)
- (under consideration): address books, bibtex, structured documents
- *Other suggestions welcome!!*

Some major open questions

- Coverage of the present tree-transformation language
  - characterization of expressive power
  - pushing the language further (binding !)
  - metatheory (type systems, algebraic theory, ...)
  - generating lenses “by example” or from schemas
- Principles of n-way synchronization
- Extending the framework to other data structures
  - dags (underway)
  - relations, ordered lists, sets, bags, etc., etc.
- Relation between trace-based and state-based (and timestamp-based, vector-clock-based, etc.) synch.