Harmony is a framework that can be instantiated to build synchronizers for a wide variety of heterogeneous, tree-structured data. Building on the system's core implementation we have assembled synchronizers for address books, browser bookmarks, calendars, structured text, and several others. Harmony's design emphasizes the use of lenses to transform the replicas before (and after) synchronization. Lenses facilitate synchronization of heterogeneous data and also simplify the synchronizer because the replicas are pre-processed (and generally made less complicated). The synchronization algorithm is simple: it traverses the replicas, merging non-conflicting updates and issuing a local schema test. This local check is enough to ensure that the results are globally well-formed.

### SYNCHRONIZATION ALGORITHM

\[
\text{sync}(S, O, A, B) = \\
\begin{cases} 
\text{equal replicas: done} & \text{if } a = b \\
\text{-- no change to } a & \text{if } b = o \\
\text{-- unresolvable conflict} & \text{if } a = b \\
\text{-- deleted more than } b & \text{if } b = \_ & \text{and } a = o \\
\text{-- create conflict} & \text{if } b = \_ & \text{and } a = (a, a) \\
\text{-- deleted more than } a & \text{else if } b = \_ & \text{and } a = (a, a); \\
\text{-- proceed recursively} & \text{else}
\end{cases}
\]

\[
V \in \text{dom}(a), \text{dom}(b) \text{ in}
\]

\[
\begin{cases} 
(\text{dom}(a) \notin \text{dom}(S) \text{ or } \text{dom}(b) \notin \text{dom}(S)) & \text{then } (X, a, b) \text{ -- schema domain conflict} \\
\text{else } (a, a') & \text{let } (o'(k), a'(k), b'(k)) = \text{sync}(S(a(k), a(k), b(k))) \\
\end{cases}
\]

### LENSES, SYNTACTICALLY

Focal is a language of combinators where every well-typed expression denotes a well-behaved lens. Every Focal program can be run forwards (get) and backwards (putback) with meaningful results in both directions. These are just a few of the interesting lenses in Focal:

- **id**: the identity;
- **compose**: puts two lenses in sequence;
- **const**: the constant lens;
- **hoist/plunge**: adds or removes an edge near the root;
- **fork/xfork**: splits the tree in two and applies a different lens to each part;
- **map/wmap**: applies a lens below every child;
- **acond/gccond**: conditionally selects a lens to apply;
- **\mu** defines recursive lenses.

### ABSTRACT BOOKMARK SCHEMA

Browser bookmark data comes in many formats (e.g., Mozilla Firefox uses HTML, Safari XML, and IE the filesystem). The lenses in our bookmark synchronizer map each of these formats to trees belonging to a common abstract schema:

```
schema Link = ('name='Value, 'url='Value)
schema Folders=('name='Value, 'contents='Contents) and Contents=List.T (Folder | ('link='Link))
schema Abstract=('bookmark='Contents, 'toolbar='Contents)
```

### MOZILLA FIREFOX LENS

```javascript
module Mozilla =

let item = lens =
  \acond ("FHT" = Any) ("separator" = o)

let subtree = lens =
  \acond ("separator" = "") ("separator" = o)

let get = lens =
  \acond ("separator" = "") ("separator" = o)

let put = lens =
  \acond ("separator" = "") ("separator" = o)
```