State Delta Resolution Algorithm

Erik Johnston

April 2018

We first define some basic functions that correspond to the synapse code.

Let K be the set of all type/state key tuples and E the set of all events. We can then define arbitrary functions:

$$\begin{aligned} f &: F \to E \\ g &: G \to E \end{aligned} \tag{1}$$

which we call state maps, for $F, G \subset K$.

We can then compute the set of all "unconflicted events":

$$U_{f,g} = \{x \mid \forall x \in F \cap G,\ f(x) = g(x)\} \cup (F \triangle G) \tag{2}$$

i.e. the set of state keys where f and g don't conflict. Similarly, we define:

$$\begin{aligned} u_{f,g} &: U_{f,g} \longrightarrow E \\ x &\longmapsto \begin{cases} f(x), & \text{if } f \in F \\ g(x), & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

which gets the unconflicted event for a given state key.

We can also define a function on $C_{f,g} = F \cup G \setminus U_{f,g}$:

$$c_{f,g} : C_{f,g} \to E \tag{4}$$

which is used to resolve conflicts between f and g. Note that $c_{f,g}$ is either $f(x)$ or $g(x)$.

Now we define:

$$\begin{aligned} r_{f,g} &: F \cup G \longrightarrow E \\ x &\longmapsto \begin{cases} u_{f,g}(x), & \text{if } x \in U_{f,g} \\ c_{f,g}(x), & \text{otherwise} \end{cases} \end{aligned} \tag{5}$$

which we call the resolved state of f and g.

Lemma 1. $\forall x \in U_{f,g}$ *s.t.* $g(x) = g'(x)$ *then* $r_{f,g}(x) = r_{f,g'}(x)$

We define

$$\alpha : E \to \mathbb{P}(K) \tag{6}$$

to be the mapping of an event to the type/state keys needed to auth the event, and

$$\alpha_{f,g}(x) = \alpha f(x) \cup \alpha g(x) \tag{7}$$

which is the set of auth events required for $f(x)$ and $g(x)$. Note that $\alpha_{f,g}(x) \subset F \cup G$.

Further, we can define

$$a_{f,g}(x) = \bigcup_{n=0}^{\infty} (\alpha_{f,g})^n(x) \tag{8}$$

to be the auth chain of $f(x)$ and $g(x)$. This is well defined as there are a finite number of elements in $F \cup G$ and $a_{f,g} \to F \cup G$.

If we consider the implementation of $c_{f,g}$ in Synapse we can see that it depends not only on the values of x, but also on the resolved state of their auth events, i.e. $r_{f,g}(\alpha_{f,g}(x))$. By "depends on" we mean that if those are the same for different values of f and g, then the result of $c_{f,g}(x)$ is the same.

Lemma 2. $c_{f,g}$ *depends only on* $a_{f,g}(x)$

Proof. $c_{f,g}(x)$ depends on $x \in a_{f,g}(x)$, and $r_{f,g}(\alpha_{f,g}(x))$. Now:

$$r_{f,g}(\alpha_{f,g}(x)) \ = u_{f,g}(\alpha_{f,g}(x)) \cup c_{f,g}(\alpha_{f,g}(x))$$

but by definition $u_{f,g}(\alpha_{f,g}(x))$ depends only on $\alpha_{f,g}(x)$, so $r_{f,g}(\alpha_{f,g}(x))$ depends on $a_{f,g}(x)$ and $c_{f,g}\alpha_{f,g}(x)$.

By induction, $c_{f,g}\alpha_{f,g}(x)$ depends on $a_{f,g}(x)$ and $c_{f,g}(\alpha_{f,g})^n(x), \forall n$. Since $(\alpha_{f,g})^n(x)$ repeats and we know $c_{f,g}$ is well defined, we can infer that $c_{f,g}(x)$ depends only on $\bigcup_{n=0}^{\infty}(\alpha_{f,g})^n(x) = a_{f,g}(x)$.

□

By inspecting the actual implementation of α we can define $a_{f,g}^{-1}(x)$ to be a function which $\forall x$, $x \in a_{f,g}^{-1}a_{f,g}(x)$. We can similarly define $\alpha_{f,g}^{-1}(x)$. Note that $\forall x$, $x \in a_{f,g}^{-1}(x)$

We now consider $g' : G' \to E$, where $g(x) = g'(x)$ except for $x \in G_\delta$, i.e. g' is a state map based on g.

Lemma 3. *For f, g, g' s.t. $\forall x \notin G_\delta,\ g(x) = g'(x)$, then $\forall x \notin a^{-1}_{f,g'}(G_\delta),\ r_{f,g}(x) = r_{f,g'}(x)$.*

Proof. Let x be s.t. $r_{f,g}(x) \neq r_{f,g'}(x)$:

$$\begin{aligned}
&\Rightarrow a_{f,g}(x) \neq a_{f,g'}(x) \\
&\Rightarrow \exists y \in a_{f,g}(x) \text{ s.t. } g(y) \neq g'(y) \\
&\Rightarrow y \in G_\delta \\
&\Rightarrow a^{-1}_{f,g'}(y) \subseteq a^{-1}_{f,g'}(G_\delta) \\
&\Rightarrow x \in a^{-1}_{f,g'}(G_\delta)
\end{aligned}$$

□

Corollary 4. *For f, g, g' s.t. $\forall x \notin G_\delta$, $g(x) = g'(x)$, then $\forall x \notin C_{f,g} \cap a^{-1}_{f,g'}(G_\delta)$, $r_{f,g}(x) = r_{f,g'}(x)$*

Proof. This follows from the previous result and that if $x \in U_{f,g} \setminus G_\delta$ then $r_{f,g}(x) = r_{f,g'}(x)$. □

This allows us to reuse most of the results of $r_{f,g}$ when calculating $r_{f,g'}$ if G_δ is small. In particular we can calculate the delta between the two functions without having to inspect $U_{f,g}$, which dramatically cuts down the amount of data used to compute deltas of resolved state of large state maps.

However, we can do better than this. We can note that $r_{f,g}(x)$ only depends on $r_{f,g}(\alpha_{f,g}(x))$ for values of $\alpha_{f,g}(x)$ not in $U_{f,g}$. Concretely, this means for example that if G_δ includes the membership of the sender of a power level event, but the power level event is in $U_{f,g}$, then we don't need to recalculate all conflicted events—despite the membership event being in every event's auth chain.

Lemma 5. *$\forall x$ s.t. $r_{f,g}(x) \neq r_{f,g'}(x)$ then $\exists y_1, ..., y_n$ s.t. $y_n \in G_\delta$, $y_i \notin U_{f,g}$ and $y_{i+1} \in \alpha_{f,g'}(y_i)$*

Proof. If $r_{f,g}(x) \neq r_{f,g'}(x)$ then $\exists y \in G_\delta$ s.t. $y \in a_{f,g'}(x)$. By definition of $a_{f,g'}(x)$, $\exists y_0, ..., y_n$ s.t. $y_0 = x$ and $y_{i+1} \in \alpha_{f,g'}(y_i)$.

We know that $r_{f,g'}(x)$ depends on either $u_{f,g'}(x)$ or $c_{f,g'}(x)$, but if $x \in U_{f,g'}$ then there is no dependency on x's auth events and so $y_n = y_0 = x \in G_\delta$. Otherwise, we have $c_{f,g}(x) \neq c_{f,g'}(x)$, which depends on $f(x)$, $g'(x)$ or $r_{f,g'}(\alpha_{f,g'}(x))$. If $x \notin G_\delta$ then we know $f(x)$ and $g'(x)$ are the same, and so $r_{f,g}(\alpha_{f,g'}(x)) \neq r_{f,g'}(\alpha_{f,g'}(x)) \Rightarrow \exists y_1 \in \alpha_{f,g'}(x)$ s.t. $r_{f,g}(y_1) \neq r_{f,g'}(y_1)$.

Applying the above to y_1 then if $y_1 \in U_{f,g'} \Rightarrow y_1 = y_n \in G_\delta$. By induction $y_i \notin U_{f,g'}$ for $i < n$.

Note that we can assume $y_i \notin G_\delta$ as otherwise we would pick $n = i$, and so if $y_i \notin U_{f,g} \Leftrightarrow y_i \notin U_{f,g'}$ □

We can use this approach and create an iterative algorithm for computing the set of state keys that need to be recalculated:

Algorithm 1 Calculate state keys needing to be recalculated

```
to_recalculate ← empty set of state keys
pending ← G_δ
while pending is empty do
  x ← pop from pending
  if x ∉ U_{f,g} then
    add all in α^{-1}_{f,g'}(x) to pending
    add x to to_recalculate
  end if
end while
return to_recalculate
```