Code Navigation with DXR

Jonas Finnemann Jensen

Mozilla

August, 2012

(Watch this presentation at air.mozilla.org)

Outline

- Introduction
- Index Optimized String Matching [1]
- Search Semantics
- Demo
- Questions

[1] Boring technical details for nerds, you've been warned.

Introduction to DXR

DXR is a web based source code index, featuring:

- Syntatic code search (substrings, regular expressions)
- Semantic code search (find method, subclasses, etc).
- Cross referencing (Jump to...)
- File and directory listings

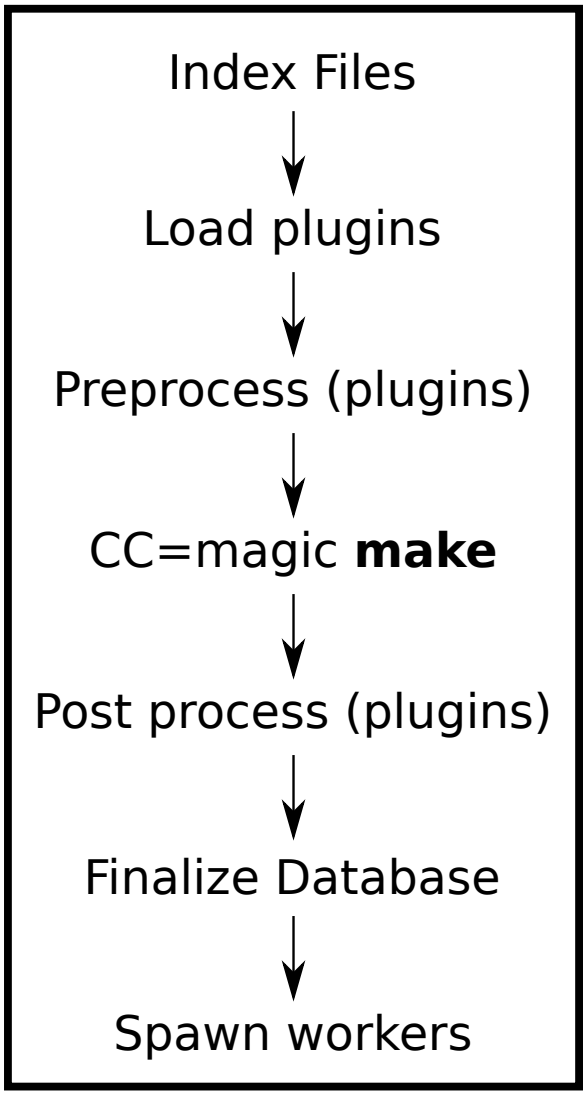
DXR is a replacement for MXR.

DXR Architecture Overview

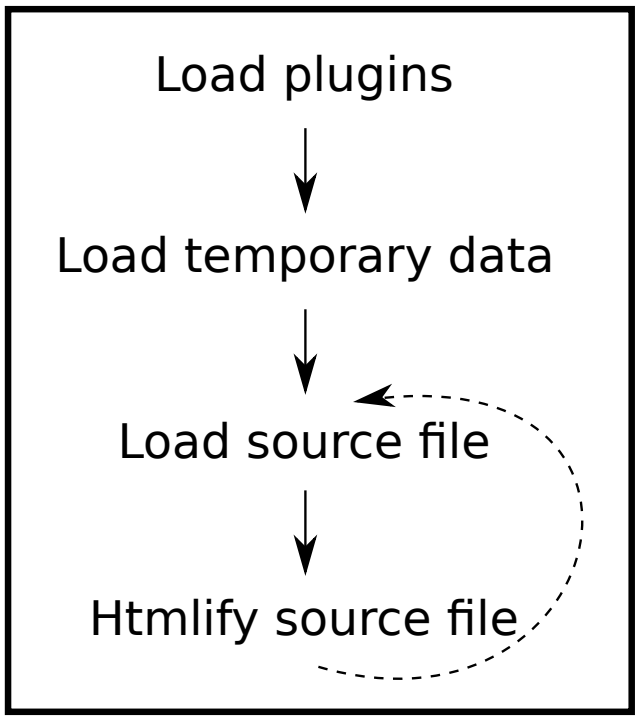
(We build HTML and database indexes offline).

DXR Build Process

./dxr-build.py --file ...



./dxr-worker.py ...




```
$ hg clone mozilla-central
$ ./dxr-build --file dxr.config
$ tar -czf output.tar.gz ...
$ scp output.tar.gz ...
```

Index Optimized String Matching

Problem

`...WHERE text LIKE '%...%'` and `...WHERE text REGEXP '...'` requires a **full table scan** in Sqlite.

Enter trigrams...

Definition (Trigram)

Given a text T a trigram of T is any substring of 3 characters in T.

e.g. $\mathit{trigrams}(\text{'abcd'}) = \{\text{'abc'}, \text{'bcd'}\}$.

Solution

We create an index as a mapping $\mathit{index} : \mathit{trigram} \rightarrow \mathit{doclist}$, from trigrams to documents (ids).

To search for substring S we scan documents in the intersection C of doclists for trigrams of S.

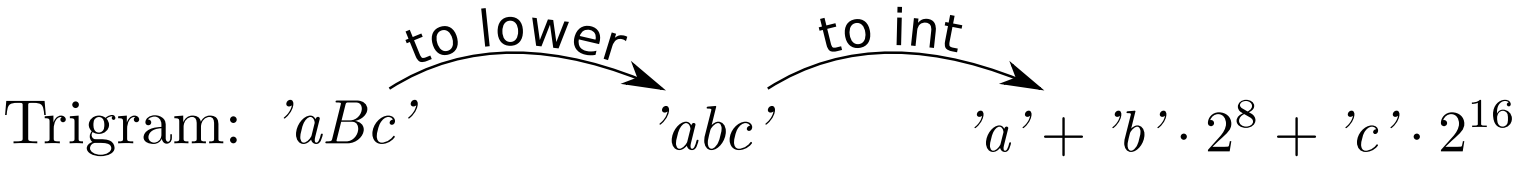
$$C = \bigcap_{t \in \mathit{trigrams}(S)} \mathit{index}(t)$$

Notice that we still need to scan the text of every document in C.

ie. an inverted index using trigrams.

The Inverted Index

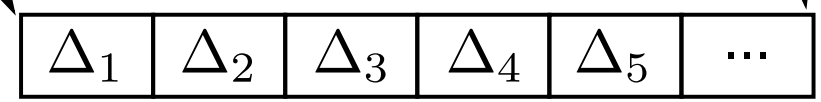
Trigram integer encoding:



Index b-tree (managed by sqlite)

```
CREATE TABLE %s_index (
  trigram INTEGER PRIMARY KEY,  /* Alias for rowid */
  doclist BLOB                  /* Binary blob */
);
```

Delta List Encoding

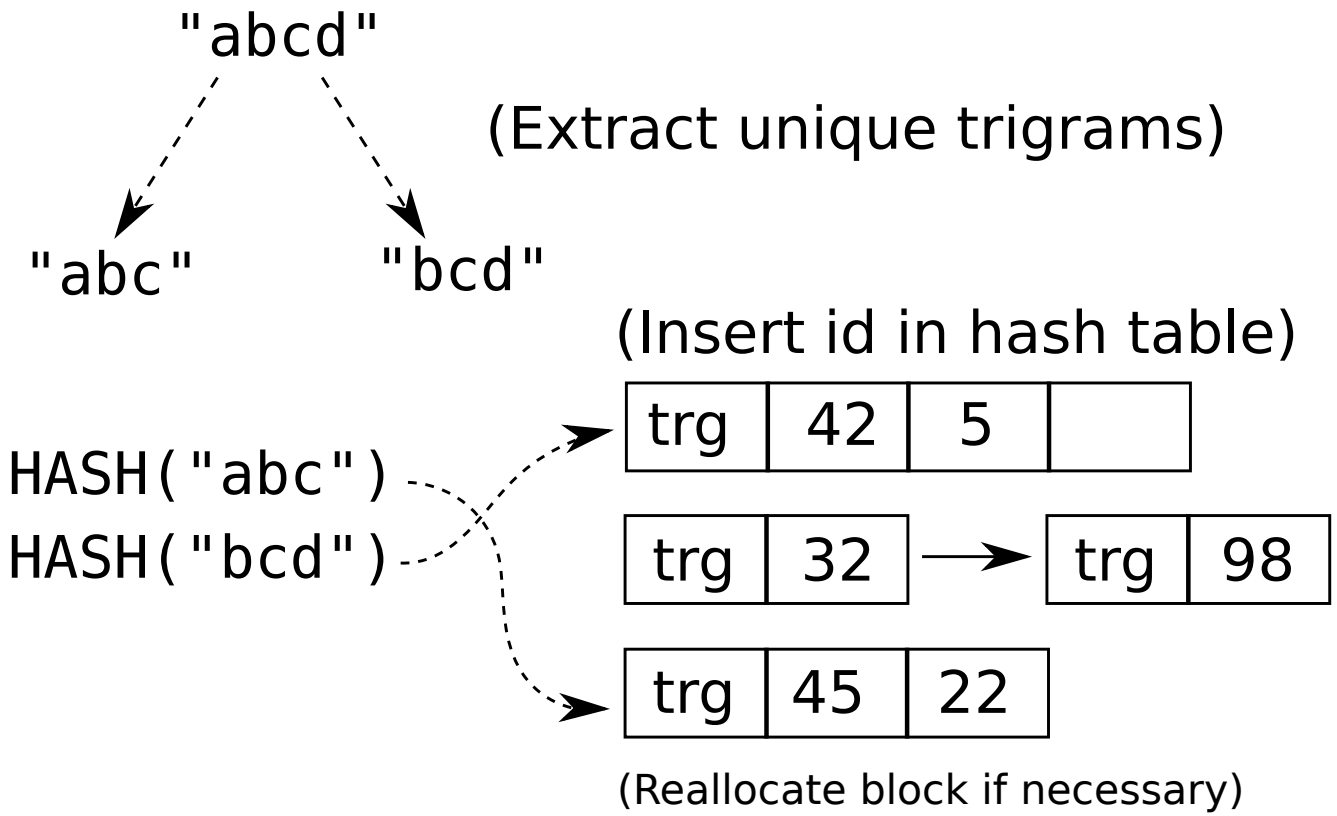


(Each Δ uses a VarInt encoding)

$$id_n = \sum_{i=0}^{n} \Delta_i$$

Building the Inverted Index

INSERT



COMMIT

Flush hash table to b-tree (ie. `%s_index`)

Extracting Trigrams From Regular Expressions

Goal: Extract required trigrams.

Methods:

(1) Recursive analysis of regular expression[2]
(How Google Code Search Worked)

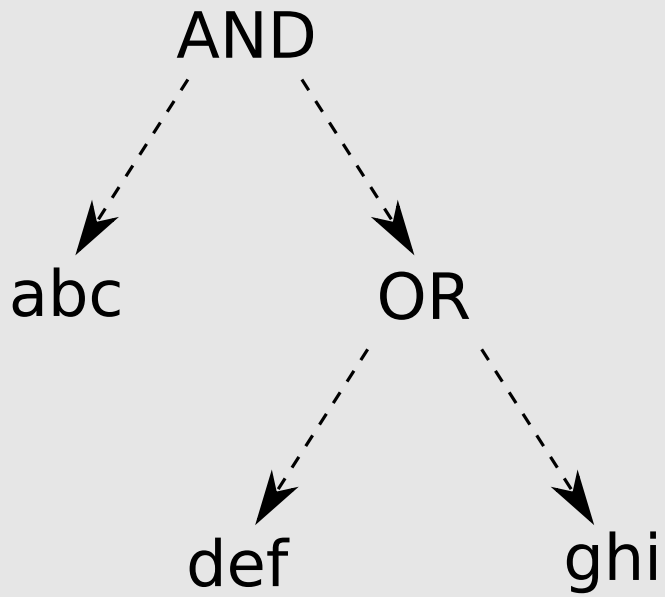
(2) Analysis of underlying automaton[3]
(As in recent WIP patch for PostgreSQL)

TriLite uses (1) as implemented by re2[4].

(An RE engine by guy who did Code Search)

Example

Regexp: `/abc(def|ghi)/`

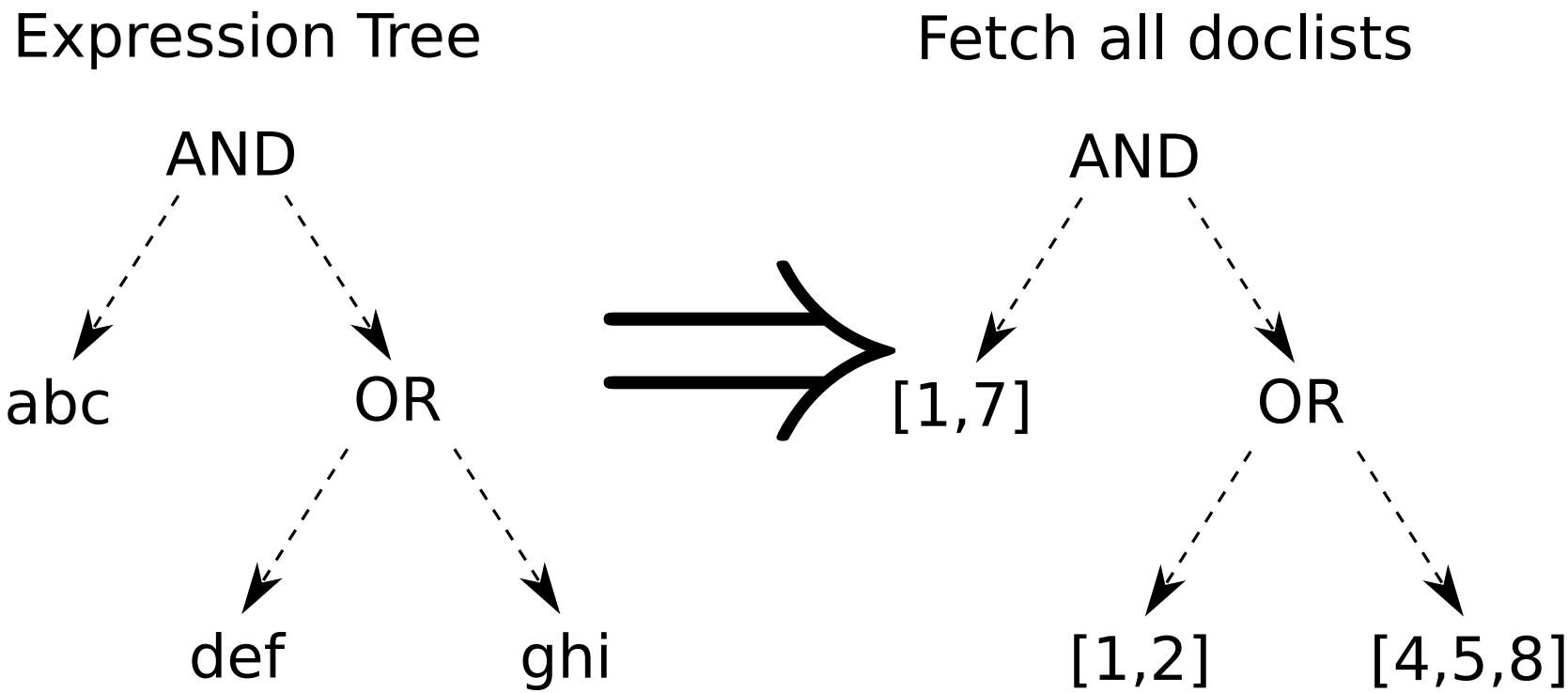


[2] http://swtch.com/ rsc/regexp/regexp4.html

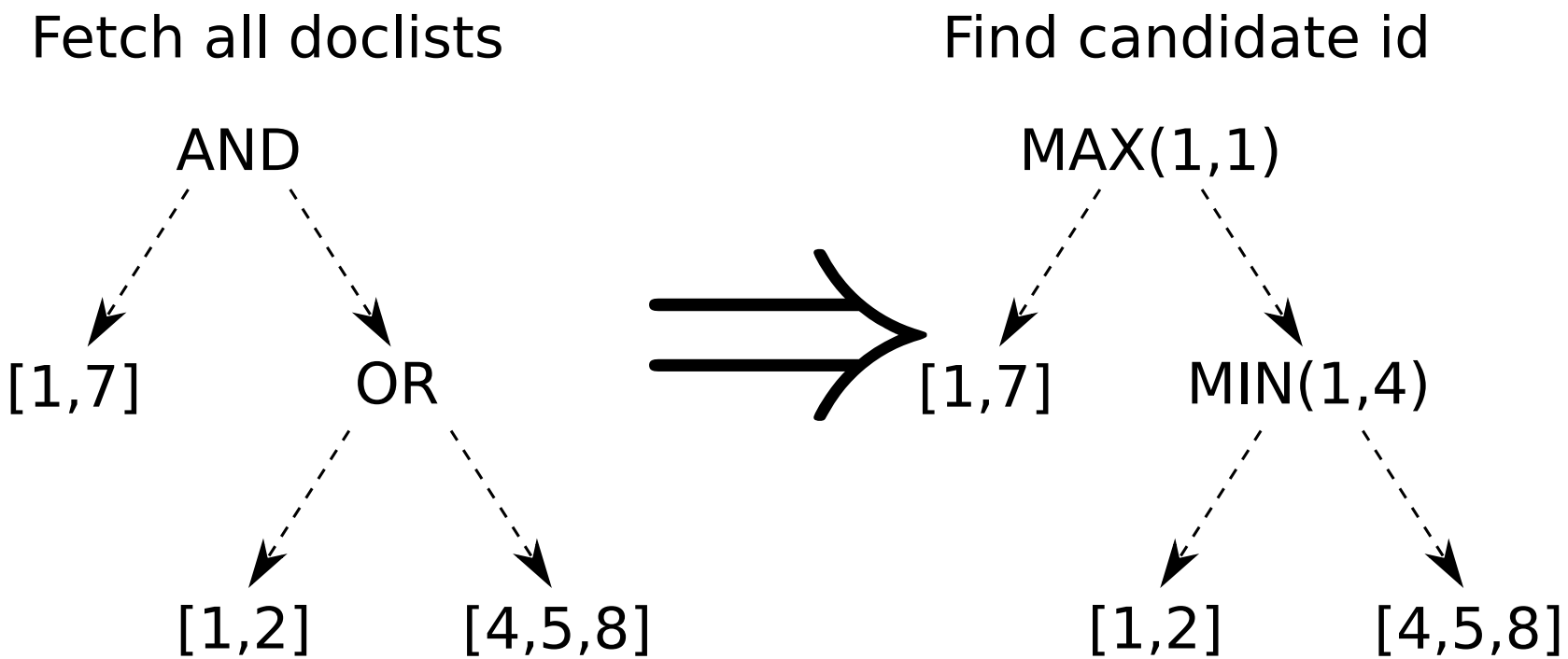
[3] http://www.pgcon.org/2012/schedule/events/383.en.html

[4] http://code.google.com/p/re2/

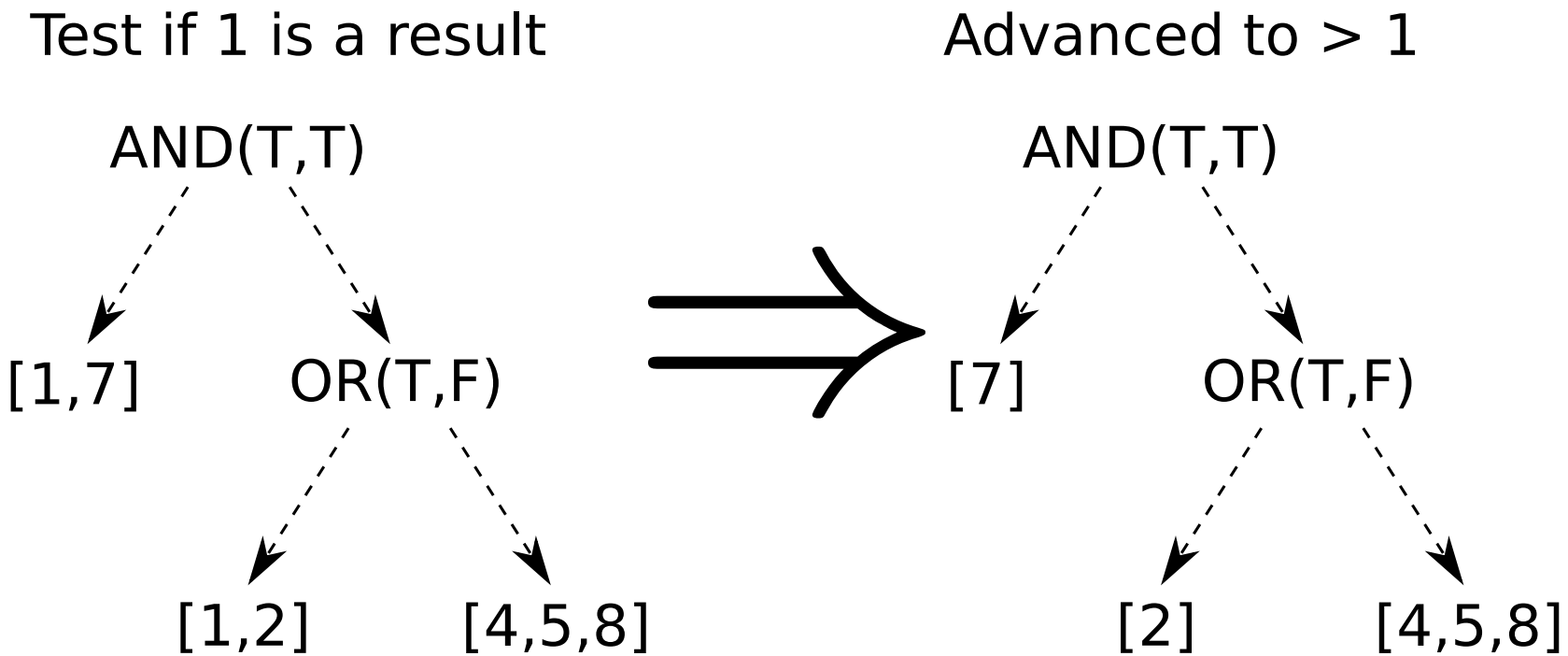
Merging DocLists (1)



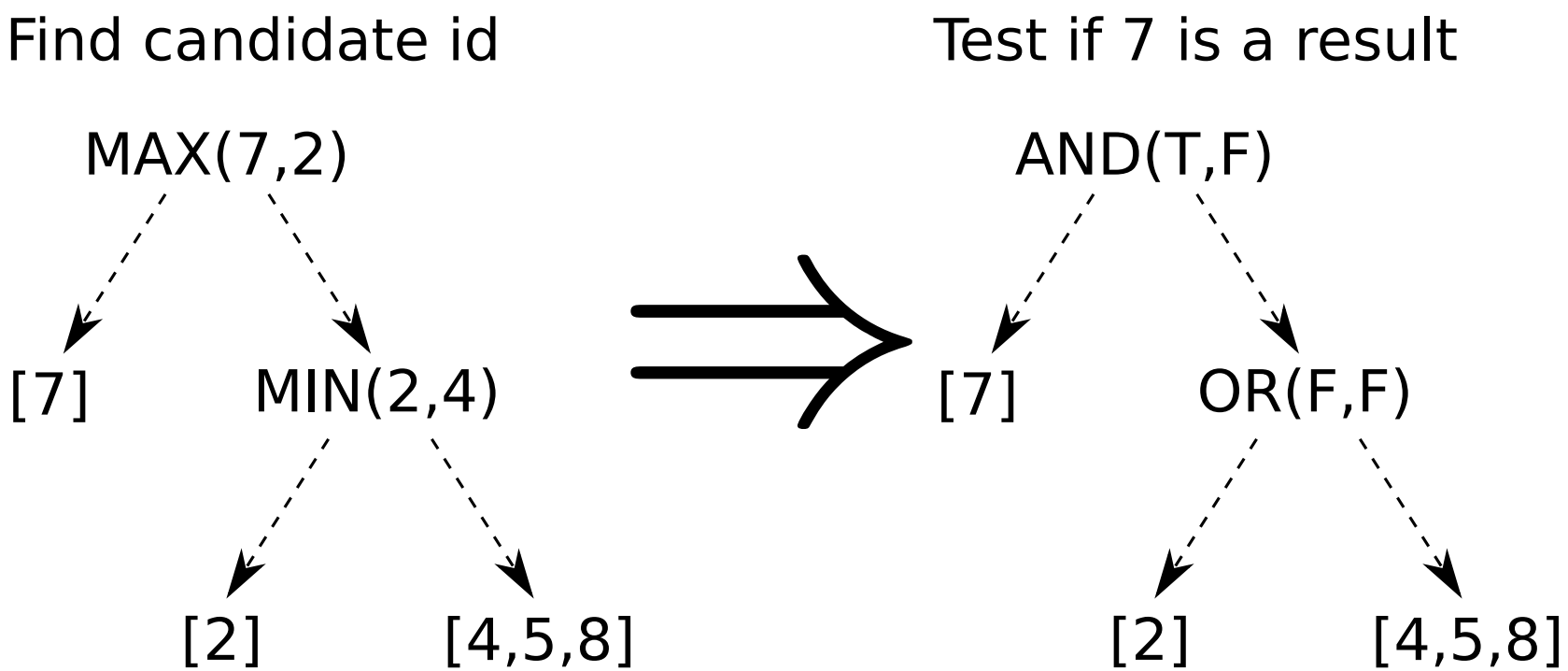
Merging DocLists (2)



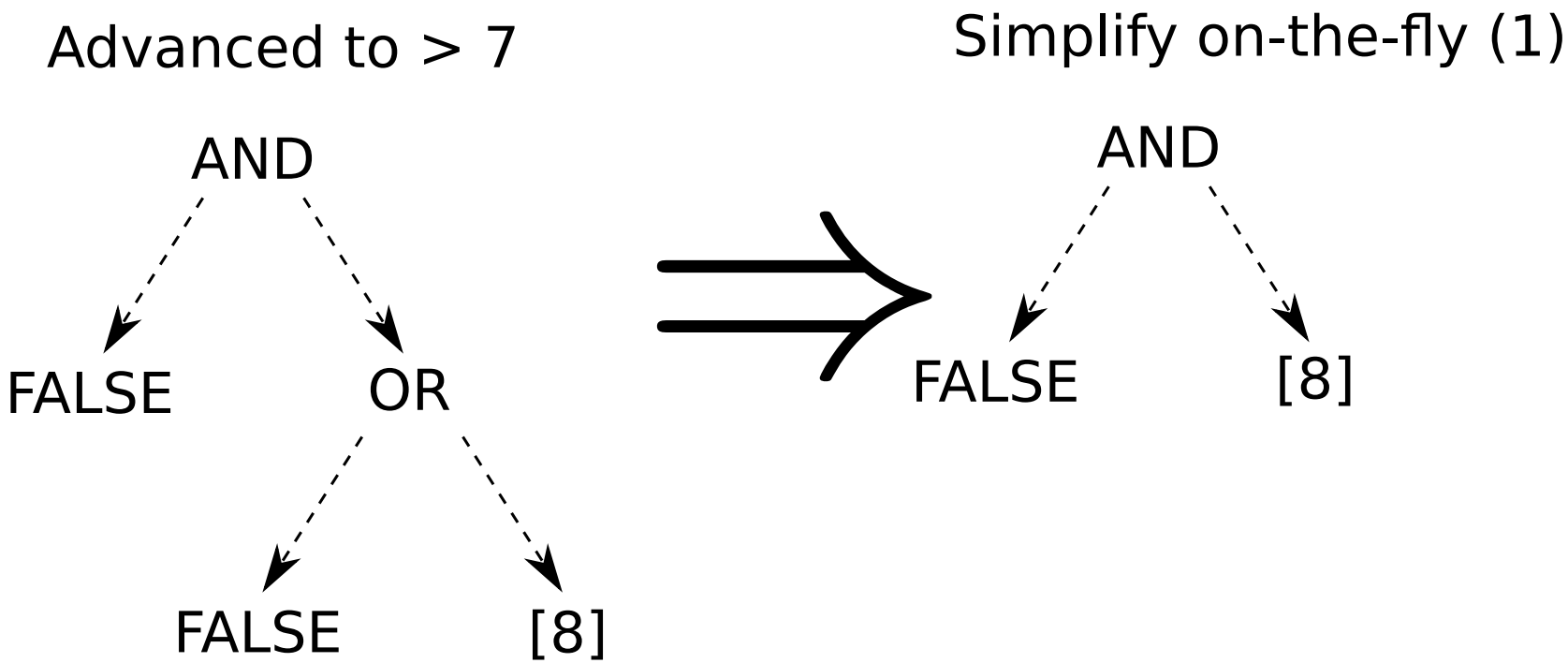
Merging DocLists (3)



Merging DocLists (4)



Merging DocLists (5)



Merging DocLists (6)

Simplify on-the-fly (2)

FALSE

This approach allows us to

- Skip more than one id at the time
- Terminate when there's no more solutions
- Merge doclists on-the-fly

TriLite Module for Sqlite

TriLite Features:

- Substring matching
- Regular expression matching
- Return start/end offsets of matches
- Scans the text after joins
- Protection from evil regular expressions.

TriLite is still in development, and **not ready** for production.

Search Semantics

Results $\subseteq$ Documents

(Only lines matched are returned)

Substrings are case sensitive.
Space denotes **AND**.
Dash **negates** a term (or argument).
Quotes searches for substrings with space.

Special Parameters

regexp	`regexp:/nsZip.*/`
path	`path:startupcache/`
ext	`ext:cpp`
type	`type:ptr`
function	`function:draw`

For more see advanced query options.

Note: `regexp:` works like vim, pattern must start and end with the same character.

Example: `"int main" Hello regexp:#(W|w)orld# -ext:h -ext:js`

Demonstration

dxr.allizom.org

(allizom, mozilla spelled backwards)

Questions

Ask away or drop by

#static

at irc.mozilla.org

DXR: https://github.com/mozilla/dxr
TriLite: https://github.com/jonasfj/trilite