Semantically Meaningful S-expression Diff

Arun Isaac

Declarative and Minimalistic Computing Devroom, FOSDEM 2021

February 6 & 7, 2021
Lisp code is data
Trivial to parse and manipulate

(define (factorial n)
  (if (zero? n)
      1
      (* n (factorial (- n 1)))))

The source is almost literally the abstract syntax tree (AST)
Automated source manipulation tools can be written easily

sdiff—a diff program for S-expressions.
Lisp code is data
Trivial to parse and manipulate

```
(define (factorial n)
  (if (zero? n)
      1
      (* n (factorial (- n 1)))))
```
Lisp code is data
Trivial to parse and manipulate

\[
(\text{define } (\text{factorial } n)
(\text{if } (\text{zero? } n)
1
(* n (\text{factorial } (- n 1)))))
\]

- The source is almost literally the abstract syntax tree (AST)
Lisp code is data
Trivial to parse and manipulate

\[
\text{define (factorial n)}
\]
\[
\text{(if (zero? n)}
\]
\[
1
\]
\[
(* n (factorial (\text{\textbf{-} n 1)})))
\]

- The source is almost literally the abstract syntax tree (AST)
- Automated source manipulation tools can be written easily
Lisp code is data
Trivial to parse and manipulate

\[
\text{(define (factorial n)} \\
\quad (\text{if (zero? n)} \\
\quad \quad 1 \\
\quad \quad (* n (factorial (- n 1)))))
\]

- The source is almost literally the abstract syntax tree (AST)
- Automated source manipulation tools can be written easily
- sdiff—a diff program for S-expressions.
The Unix world of lines
Files are a flat list of lines
Thanks to Unix legacy, most shell utilities (sed, grep, awk, cut, etc.) operate on lines.
The Unix world of lines
Files are a flat list of lines

- Thanks to Unix legacy, most shell utilities (sed, grep, awk, cut, etc.) operate on lines.
- For example, GNU Diff outputs the difference as a list of lines to be inserted and deleted.
The Unix world of lines
Files are a flat list of lines

- Thanks to Unix legacy, most shell utilities (sed, grep, awk, cut, etc.) operate on lines.
- For example, GNU Diff outputs the difference as a list of lines to be inserted and deleted.

@@ -7,5 +7,5 @@
((? string?)
  (updated-url source-uri))
((source-uri ...)
-(find updated-url source-uri))))
+(any updated-url source-uri))))
(_ #f))
Lisp projects use diff too

Impedance mismatch between S-expressions and line-oriented diff
Can you spot the actual change in the following diff?

```
< (/ (+ (- b)
<   (sqrt (- (* expt b 2)
<     (* 4 a c)))))
<   (* 2 a))
---
> (let ((b 1))
>   (/ (+ (- b)
>     (sqrt (- (* expt b 2)
>       (* 4 a c)))))
>   (* 2 a)))
```
We need a tree diff for S-expressions

Not a line diff

We need a tree diff.

(let ((b 1))
  (/ (+ (- b)
      (sqrt (- (* expt b 2)
              (* 4 a c))))
      (* 2 a)))
Tree diff

A surprisingly difficult problem
Extracting the author’s intent from the old and new files is hard, and probably requires general AI.
Tree diff
A surprisingly difficult problem

- Extracting the author’s intent from the old and new files is hard, and probably requires general AI.
- Approximate by posing it as an optimization problem.
Tree diff
A surprisingly difficult problem

- Extracting the author’s intent from the old and new files is hard, and probably requires general AI.
- Approximate by posing it as an optimization problem.
- For unordered trees, the problem is $NP$-hard.
Tree diff
A surprisingly difficult problem

- Extracting the author’s intent from the old and new files is hard, and probably requires general AI.
- Approximate by posing it as an optimization problem.
- For unordered trees, the problem is \( \mathcal{NP} \)-hard.
- We only deal with ordered trees.

sdiff implements the MH-DIFF (Meaningful Hierarchical Diff) algorithm.
Tree diff
A surprisingly difficult problem

- Extracting the author’s intent from the old and new files is hard, and probably requires general AI.
- Approximate by posing it as an optimization problem.
- For unordered trees, the problem is $\mathcal{NP}$-hard.
- We only deal with ordered trees.
- sdiff implements the MH-DIFF (Meaningful Hierarchical Diff) algorithm.

MH-DIFF supports 6 operations—insert, delete, update, move, copy and glue.
MH-DIFF supports 6 operations—insert, delete, update, move, copy and glue

With associated costs $c_i$, $c_d$, $c_u(\text{old}, \text{new})$, $c_m$, $c_c$, $c_g$ respectively
MH-DIFF supports 6 operations—insert, delete, update, move, copy and glue

With associated costs $c_i$, $c_d$, $c_u(old, new)$, $c_m$, $c_c$, $c_g$ respectively

Posed as an optimization problem: to find an edit script such that the total cost is minimized.
MH-DIFF supports 6 operations—insert, delete, update, move, copy and glue

With associated costs $c_i$, $c_d$, $c_u(\text{old, new})$, $c_m$, $c_c$, $c_g$ respectively

Posed as an optimization problem: to find an edit script such that the total cost is minimized.

MH-DIFF operates in two phases.
MH-DIFF supports 6 operations—insert, delete, update, move, copy and glue

With associated costs $c_i$, $c_d$, $c_u(\text{old}, \text{new})$, $c_m$, $c_c$, $c_g$ respectively

Posed as an optimization problem: to find an edit script such that the total cost is minimized.

MH-DIFF operates in two phases.

1. Match old and new trees.
MH-DIFF supports 6 operations—insert, delete, update, move, copy and glue

With associated costs $c_i$, $c_d$, $c_u(old, new)$, $c_m$, $c_c$, $c_g$ respectively

Posed as an optimization problem: to find an edit script such that the total cost is minimized.

MH-DIFF operates in two phases.

1. Match old and new trees.
2. Extract an edit script from the matching.
MH-DIFF
Matching old and new trees

Match changed/unchanged parts of old and new trees.

Figure: Old tree

Figure: New tree
Begin with a complete bipartite graph with old tree nodes on one side and new tree nodes on the other.
Begin with a complete bipartite graph with old tree nodes on one side and new tree nodes on the other. An edge is a potential matching of old and new trees, and comes with a cost.
Begin with a complete bipartite graph with old tree nodes on one side and new tree nodes on the other.

An edge is a potential matching of old and new trees, and comes with a cost.

Goal: Prune edges to minimize total cost.
Begin with a complete bipartite graph with old tree nodes on one side and new tree nodes on the other.

- An edge is a potential matching of old and new trees, and comes with a cost.
- Goal: Prune edges to minimize total cost.
- The minimum cost edge cover problem can be solved using the Hungarian algorithm.
Demos!

The fun part
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
- maybe improve the cost model and support move, copy and glue operations
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
- maybe improve the cost model and support move, copy and glue operations
- fully support irregular lisp syntax such as quoting, line-based comments, etc.
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
- maybe improve the cost model and support move, copy and glue operations
- fully support irregular lisp syntax such as quoting, line-based comments, etc.
- cleaner and more concise diff output
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
- maybe improve the cost model and support move, copy and glue operations
- fully support irregular lisp syntax such as quoting, line-based comments, etc.
- cleaner and more concise diff output
- a more optimized implementation that scales better
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
- maybe improve the cost model and support move, copy and glue operations
- fully support irregular lisp syntax such as quoting, line-based comments, etc.
- cleaner and more concise diff output
- a more optimized implementation that scales better
- integrate and replace tooling such as git diff
Going forward
Plenty still needs doing!

- sdiff isn’t ready for everyday use yet
- plenty of bugs to fix and a lot more testing necessary
- maybe improve the cost model and support move, copy and glue operations
- fully support irregular lisp syntax such as quoting, line-based comments, etc.
- cleaner and more concise diff output
- a more optimized implementation that scales better
- integrate and replace tooling such as git diff
- use as diff for other S-expression data (such as LibrePCB)
Code is available under GPLv3 at
https://systemreboot.net/files/sdiff-fosdem2021.tar.gz

- Would you use sdiff?
- How can sdiff be more useful?
- Feedback and criticism welcome!

arunisaac@systemreboot.net