anagallis version 0.998 beta, 16 April 2017

First public version.
Statically linked 32-bit executable compiled with gcc 4.8.4.

anagallis version 0.999 beta, 9 September 2017

Statically linked 32-bit executable compiled with gcc 7.3.0.

Bug fix: in some cases, the reconstructed aggregate stateset at an inner node contained states in addition to the union of the non-aggregate statesets at that node.

Some minor changes in the documentation.

anagallis version 1.00, 6 November 2018

Statically linked 32-bit executable compiled with gcc 7.3.0.

Major rewrite of the section on the theory behind the program in the built-in documentation.
Addition of a section with a high level description of the algorithm, including an informal definition of non-aggregate and aggregate final statesets.
Several minor changes, additions, and corrections in the documentation.

In some cases, the reconstructed non-aggregate statesets at an inner node contained states in addition to the optimal states (as a consequence, these showed up in the aggregate statesets as well, which in turn could affect collapsing of zero-length branches). This was due to a missing re-initialization during backtracking while searching for optimal reassignments to 'presence' in initial regions of absence for subordinate root characters in character hierarchies. Fixed.

Enabled calculation of strict consensus trees.

Non-aggregate statesets are now always plotted when requested (previous versions in some cases did not plot non-aggregate statesets, with a warning to that effect).

anagallis version 1.01, 11 December 2018

First version with MacOS executable.

Linux: statically linked 32-bit executable compiled with gcc 7.3.0.
MacOS: dynamically linked executable.

Correction of some dates in this changelist.

Reading parenthetical trees with terminal names didn't work from the moment that at least one terminal had a name with a 'v', an 'o', an 'i' or a 'd'. Fixed.

Several small changes in the use of variadic functions, fixing some issues that popped up when porting the code to MacOS.

Built-in documentation:

- correction of a previous erroneous assertion that initial regions of absence can in general be optimized independently; discussion of the consequences thereof;
- addition of a section on the scope and limits of the current algorithm;
- several small changes and corrections.

anagallis version 1.02, 24 May 2020

Fixed a problem with setting/reporting defaults used for the commands that plot trees when this was done with no character data in memory (the program dumped).

Fixed a problem with plotting optimizations of character hierarchies: when a tree happened to be chopped on a node that needed the '+' placeholder to be plotted, the chopped subtree contained a spurious and dangling branch below that placeholder.

Fixed a problem with the -i option (plot large subtrees first) whenever plotting trees (it sometimes dumped).

Searches for optimal trees can look for trees of minimal score (default) or trees of maximal score. While doing tree searches, the output in both cases contained the phrases 'best score'. That is now replaced by 'min score' and 'max score'.

During the download pass in the optimization of a subordinate root character in a character hierarchy, it has to be detected when a region of presence for that character is entered. One specific case that can occur starting from a second degree subordinate root character on went undetected (it depends on an interaction between two of its higher level root characters). Each such case led to an overestimation by one of the number of indels in the region of presence that is involved. All versions - this one included - calculate the number of indels in two different ways: a complicated one that is required to calculate other required numbers, and a more straighforward one afterwards, just for verification of the initial result. Whenever there is a discrepancy between both calculations, the program immediately exits with an error message that points out the discrepancy, a crude approach to avoid reporting wrong tree scores and wrong optimal trees. Thanks to Martin Brazeau and coworkers for bringing such a case to my attention, it pointed to this issue with the download pass for subordinate root characters. A similar problem was present in the downbload pass for simple subordinate characters). Both fixed.

In some cases, reconstructed non-aggregate and aggregate statesets of subordinate characters in a character hierarchy at inner nodes still erroneously contained state zero (a subset of the cases where such statesets also contain 'inapplicability'). This did not affect reported tree scores but could affect collapsing of zero-length branches. Fixed.

It is a good idea to keep command names backwards compatible, but some command names have changed in this version to correct a spelling error. It 'to pause', not 'to pauze'. Therefore a command like 'script execute pauze' that used to be valid now triggers an error. It has to be changed to 'script execute pause'.

Anagallis 1.01 was the first version with a MacOS executable and a Linux executable. It turned out that both used a different implementation/configuration of the ISO C function rand() and srand(). As a result, the MacOS and the Linux executable produced different sequences of pseudorandom numbers when provided with the same seed. Given that such sequences are for example used to determine random addition sequences in tree builds, tree searches with identical settings could result in different trees between both executables. To make sure that results are fully reproducible between both executables, this version uses a simple recursive pseudorandom generator that does not rely on special C functions and that still produces sufficient randomness for its indended use. The drawback is that the current executables both behave differently in this respect than their older versions. This is especially so for the command to change the current seed: zero is no longer accepted as a valid seed, seeds have to be strictly positive now, with 1 as default. For simplicity, there is no switch to revert to the old behavior.

When plotting final statestes for a root character that deviates from the default that state 0 represents absence and state 1 presence, the states that code for absence and presence are now first explicitly listed. That should make it easier to interpret such plots.

When optimizing a character hierarchy on a tree, the reported score is guaranteed to be optimal as long as the hierarchy has a whole has no region of presence with more than 11 neighbouring regions of absence on that tree. Whenever that happens, the score is no longer guaranteed to be optimal. The difference with previous versions is twofold. First, the program now better indicates when this happens: command 'characters score <tree scope>' always indicates in its output for which character hierarchies this is the case on which trees. In previous versions, this was indicated at the level of tree scores (command 'trees score <tree scope>'), not indicating for which out of possibly several character hierarchies the scores might be suboptimal. Second, in previous versions the program did a limited attempt at further optimization beyond the initial assignments of regions of absence in such cases. That further attempt at optimization is now skipped.

anagallis version 1.02a, 27 May 2020

Bug fix release after a double bug report by Martin Brazeau.

The first had do with proper treatment of '?' for missing data in main and subordinate root characters of character hierarchies while calculating all possible final statesets. It sometimes led to failure of a built-in double check of calculations, making the program exit. Fixed.

The second had to do with use of an uninitialized memory location for some datasets in a fast tree build procedure that is only used to calculate the minimum possible length of characters, leading to undefined behaviour (thanks valgrind!). Awaiting a proper fix, I inactivated the command that triggers that fast build in this release, and skipped the preprocessing step that identifies uninformative regular Fitch and Farris characters outside character hierarchies (it requires that minimum possible length). Compared to the complexity of optimizing character hierarchies, the overhead of including uninformative regular Fitch and Farris characters in the score calculations is negligible.