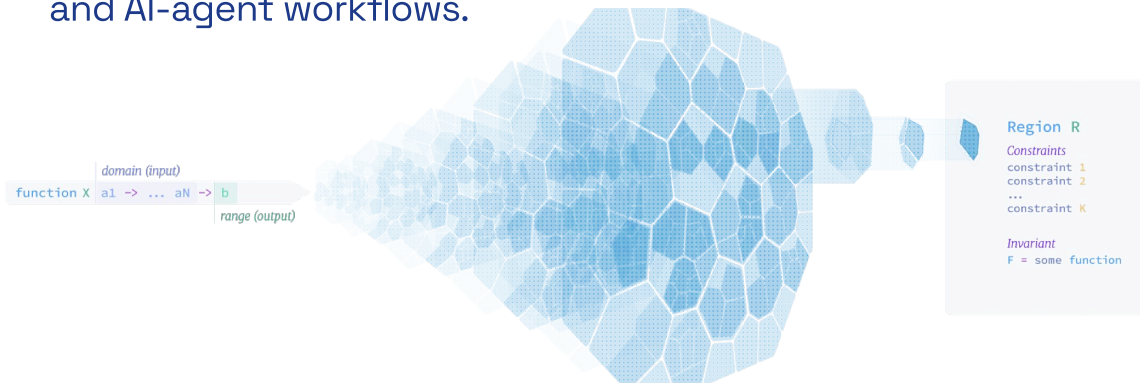


ImandraX

ImandraX is the core automated reasoning engine powering the Imandra stack.

It combines symbolic execution, novel decision procedures (lifting of SMT to a Higher-Order Logic with recursion and induction), automated and interactive theorem proving, model checking, and region decomposition to analyze software and formal models at scale.

ImandraX can be used directly through Python APIs, the VS Code extension, or integrated into higher-level products and AI-agent workflows.



```
1 Verification Goals
2 *****
3
4 #> let board
5
6 (* Now that we know the loop works for positive n, let's prove that
7 it works for any n. *)
8
9
10 X Re-check | Browse | Interact | Copy
11
12 lemma loop_correct stack ra rn s steps =
13   steps >= 11*rn + 3
14   && s = { halted=false; prog=ex; pc=2; stack; ra; rn }
15   ==>
16   run s steps
17   = (if rn > 0 then
18     { halted=true; prog=ex; pc=2; stack; ra; rn }
19     else
20     { halted=true; prog=ex; pc=2; stack; ra; rn })
21   [@@by unroll 100]
22
23
24 Counter model:
25 ...
26 model {
27   let rn : int = (-7719)
28   let steps : int = (-84898)
29   let ra : int = 1142
30   let s : state =
31     {halted = false;
32     prog =
33       [Const 0; Store RA; Load RN; If
34         Store RA; Load RN; Const 1; Sub; Store RN; Goto (-10); Halt];
35     pc = 2; stack = stack; ra = rn +
36       ((-11) + steps)
37     }
38   ==> run
39   {halted = false;
40   prog =
41     [Const 0; Store RA; Load RN; If
42       Store RA; Load RN; Const 1; Sub; Store RN; Goto (-10); Halt];
43   pc = 2; stack = stack; ra = rn + ra + sum ((-1) + rn);
44   rn = 0;
45   (steps + (-11) * rn)
46   }
47
48 C0. rn <= 0
49 C1. run s steps
50 ==> run
51 {halted = false;
52 prog =
53 [Const 0; Store RA; Load RN; If
54 Store RA; Load RN; Const 1; Sub; Store RN; Goto (-10); Halt];
55 pc = 2; stack = stack; ra = ra + sum rn; rn = 0;
56 (steps - 11 * rn)
57 }
```

```
236 lemma loop_correct stack
237   steps >= 11*rn + 3
238   && s = { halted=false;
239   && rn >= 0
240   ==>
241   run s steps
242   = (if rn > 0 then
243     { halted=true; prog=ex; pc=2; stack; ra; rn }
244     else
245     { halted=true; prog=ex; pc=2; stack; ra; rn })
246   [@@by {use loop_correct,
247   @> {use program_base,
248   0
249   {s with pc=2;
250   (steps - 11*rn)
251   @> auto]
252   [@@disable run]
```

```
165 instance (fun acts ->
166   match run dim_8x8 empty_board acts with
167   | Some covering ->
168     List.length covering = 10
169     | None -> false)
170
171 let mutilated_board =
172   delete {x=0;y=0} (delete {x=dim_8x8;y=0})
173
174 lemma mutilated_color_count =
175   num_white mutilated_board
176   <= num_black mutilated_board
177   [@@by auto]
178
179 (* Finally, our main theorem:
180 you can't cover a mutilated chessboard
181 *)
182 theorem main acts =
183   match run dim_8x8 empty_board acts with
184   | Some covering ->
185     covering <= mutilated_board
186     | None -> true
187   [@@by {use run_color_count_from_empty_board,
188   @> {use mutilated_color_count]
189   @> auto]
```

See www.imandrax.dev for more info.

CodeLogician: Logical Reasoning for AI Coding Agents

CodeLogician makes AI coding agents structure their reasoning in logic and uses ImandraX to systematically analyze what the code can actually do. So you catch edge cases, surface hidden behaviors, and ship with confidence instead of guesswork.

→ LLM + CodeLogician

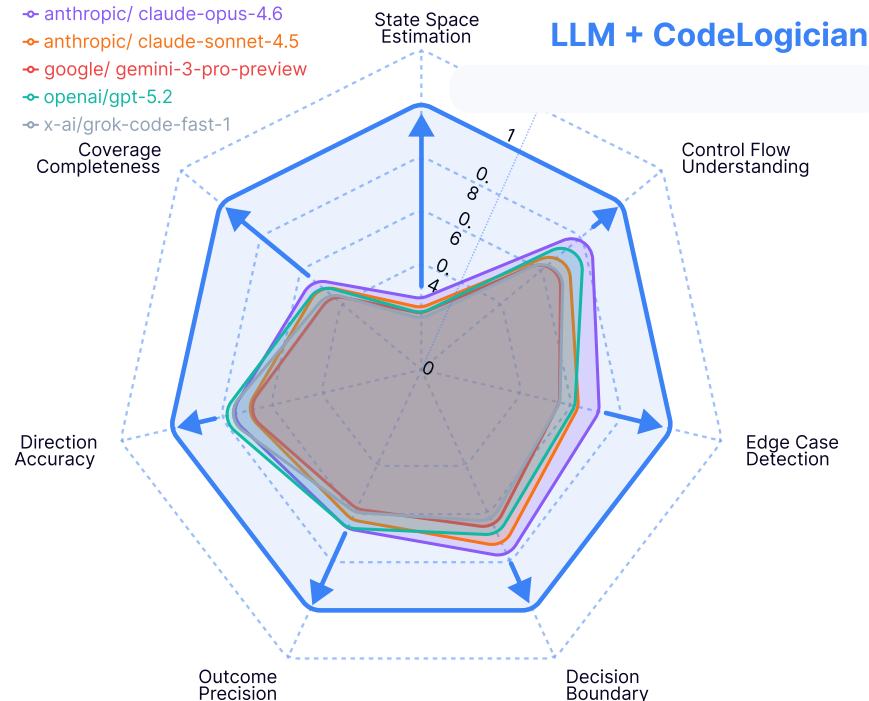
→ anthropic/ claude-opus-4.6

→ anthropic/ claude-sonnet-4.5

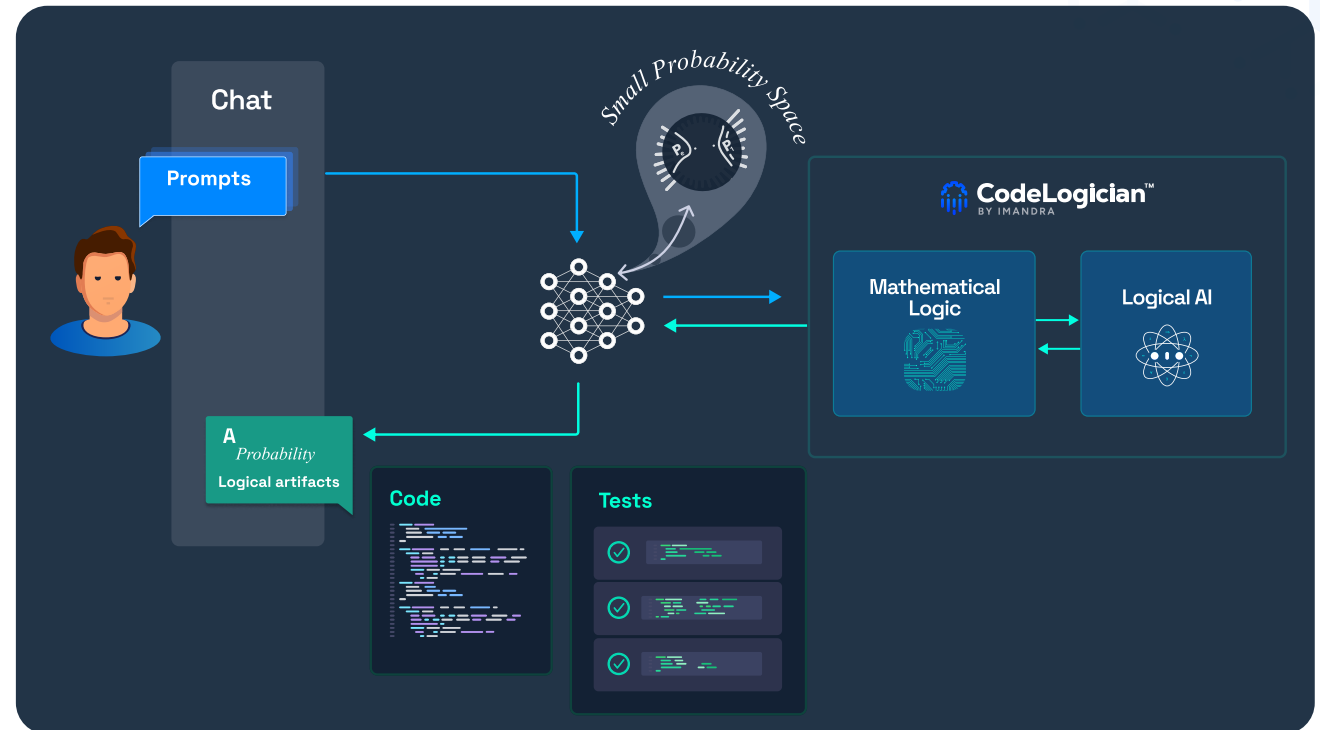
→ google/ gemini-3-pro-preview

→ openai/gpt-5.2

→ x-ai/grok-code-fast-1



With CodeLogician



- Decision logic is formalized
- Reasoning engine explores behavior
- Artifacts expose boundaries and edge cases
- State machines and workflows
- Financial and rule engines
- Access control and policy systems
- Distributed system coordination
- Complex integration logic

CodeLogician turns AI coding from “looks right” into “behavior understood.”

See www.codelogician.dev for more info.

Pones: computable governance

TraceHub is an online environment for storing, editing, analyzing, and governing reasoning traces produced by AI agents.

Instead of treating agent execution as opaque prompt logs, TraceHub structures reasoning into analyzable artifacts connected to:

- code,
- specifications,
- policies,
- verification goals,
- proofs,
- counterexamples,
- and generated tests.

Reasoning traces become:

- inspectable,
- searchable,
- enforceable,
- replayable,
- and mathematically analyzable.

Ponens enables “glass-box” governance for AI-assisted engineering workflows.

www.ponens.dev

The screenshot displays the TraceHub interface, titled "FORMAL REASONING POWERED BY IMANDRAX". It shows a sequence of six reasoning artifacts, each in a card with a title, description, and formal representation.

- #20 Formalize: updated flow with 3DS + high-risk**
extend the IML model with new features: RequiresAction status, 3DS flags, approval_count, high_risk requiring 2 approvals
TRANSPARENT
(* Fixed capture + new may_capture with 3DS/approval checks *)
let required appr...
transparent formalization -> 6 new/updated IML symbols
← bug_fix_applied
← original_iml_model
updated_iml_model →
- #21 Define: 7 amount invariant properties (one per transition)**
verify that EVERY transition function preserves the amount invariant
create preserves amounts
confirm preserves amounts
three ds preserves amounts +4 more
7 verify goals defined, one per transition function
← updated_iml_model all_amount_vgs →
- #22 Verify: all 7 amount invariants - PROVED**
after the fix, all transition functions preserve the amount invariant
PROVED
create preserves amounts
confirm preserves amounts
three ds preserves amounts +4 more
proved - all 7 amount invariants hold
← all_amount_vgs amount_invs_proved →
- #23 Define: 3DS blocks authorization until completed**
new feature property: if 3DS is required but not completed, confirm must move to RequiresAction, not Authorized
let goal_3ds_blocks_authorization (o: order) =
 o.status =
verify: 3DS gates authorization flow
← updated_iml_model 3ds_block_vg →
- #24 Verify: 3DS blocks authorization - proved**
confirmed: 3DS challenge properly gates the authorization flow
PROVED
proved by T_deduction
← 3ds_block_vg 3ds_block_proved →
- #25 Define: high-risk needs 2 approvals before capture**
new feature property: high-risk orders with fewer than 2 approvals must not reach Captured status
let
 goal_high_risk_needs_two_approvals (amt: int) (o: order) =
 verify: high-risk capture requires 2 approvals
← updated_iml_model high_risk_vg →

At the bottom, a summary artifact is shown:

- reasoning_required_for_high_stakes_changes** **PASSED**
Edits to payment or risk modules must be justified by prior formal reasoning artifacts in the dependency chain
MODULE REASONING_REQUIREMENT
EditFile action 5 has proved verification results (a10, a16) in its dependency chain
Evidence: actions #22, #28
Evidence: artifacts a10, a16
SCOPE
action_type: EditFile path_matches_any: payments/**, risk/**, stripe_payment_flow.py
FORMULA
G EditFile A high_stakes_path → Pchain VerificationResult(proved v sat) v Decomposition()

Imandra Protocol Language

Imandra Protocol Language (IPL) is a formal DSL for expressing complex financial APIs, workflows, and protocol behavior.

IPL models define: message schemas, validity conditions, state transitions, response logic, internal events, and operational constraints.

IPL models compile into IML and become analyzable by ImandraX.

```
74 // These are not required within the orders, etc...
75 filledQty : Qty = 0.0;
76 current_time : UTCTimeOnly = UTCTimeOnly (8, 0, 0, None);
77
78 // Define the opening and closing times
79 opening_time : UTCTimeOnly = UTCTimeOnly (8, 0, 0, None);
80 closing_time : UTCTimeOnly = UTCTimeOnly (16, 0, 0, None);
81
82
83 // Define how the model should process a validated
84 // incoming NewOrderSingle message
85 receive ( msg : NewOrderSingle ) {
86
87   // Extract all relevant fields from within the
88   assignFrom (msg, state) {
89
90     let x = state.Price
91
92     if x == 0.0 then return
93     return
94
95     state.leavesQty = msg.leavesQty;
96     state.live_order = true;
97
98     send ExecutionReport {
99       state with
100         ExecID = "";
101         ExecType = ExecType.New;
102     }
103   }
104
105 // Handle rejections for 'NewOrderSingle'
106 reject { msg : NewOrderSingle, rejectReason : ... }
```

```
17 // Side can reference the union of values from
18 // such as Redeem
19 req Side valid when it != Side.Redeem
20 req TransactT This message is assumed to be from
21 req OrdType v e from FIX_4_2; to avoid ambiguity
22 req HandlInst name for example "message FIX_4_
23 req Symbol
24
25 FIX message tag: 8; The execution re
26 Confirm the receipt of an order 2
27 order (i.e. accept cancel and rep
28 status information 4. Relay fill
29 Relay fill information on tradeab
30 6. Reject orders 7. Report post-
31 with a trade
32
33 validate { st
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

