西安电子科技大学 XIDIAN UNIVERSITY | 广州研究院 Guangzhou institute of technology

大模型驱动的形式化定理证明：综述与展望

&

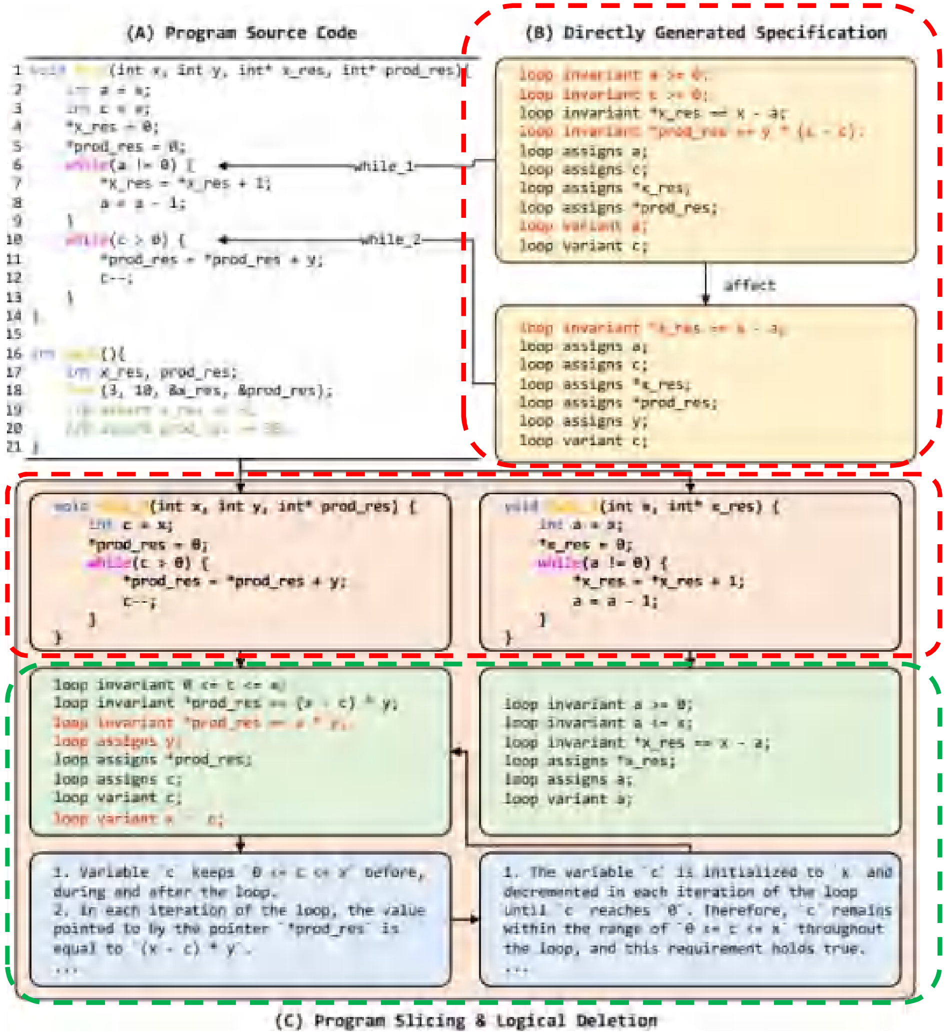
SLD-Spec: Enhancement LLM-assisted Specification Generation for Complex Loop Functions via Program Slicing and Logical Deletion

胡俊杰

2025 Sep 25

SLD-Spec:

Enhancement LLM-assisted Specification Generation for Complex Loop Functions via Program Slicing and Logical Deletion



1、一次性给**LLM**太多的上下文信息，导致该节点会生成其他节点的规约内容。—— ***Program Slicing***

```
void func(){
    //>>INFILL<<
    while-1;
    …
    while-2;
}
```

Slicing →

```
void func1(){
    >>INFILL<<
    while-1;
   >>END<<
    …
}
```

&

```
void func2(){
    >>INFILL<<
    while-2;
    >>END<<
    …
}
```

2、验证器发现当前节点生成的规约是错的？不一定，可能是缺乏前置条件或者要等到其他节点的规约生成后才能验证。

—— ***Logical Deletion:*** 让LLM自己推理放在此处合不合理，而不去调用验证器执行非0即1的验证。

Fig. 1. A motivation example from the Complex-Loop benchmark. (A) Source code for the example program. (B) Loop specifications automatically generated by AutoSpec for the program. (C) Workflow of SLD-Spec applied to the example. Specifications that fail verification are highlighted in red.

SLD-Spec:
Enhancement LLM-assisted Specification Generation for Complex Loop Functions via Program Slicing and Logical Deletion

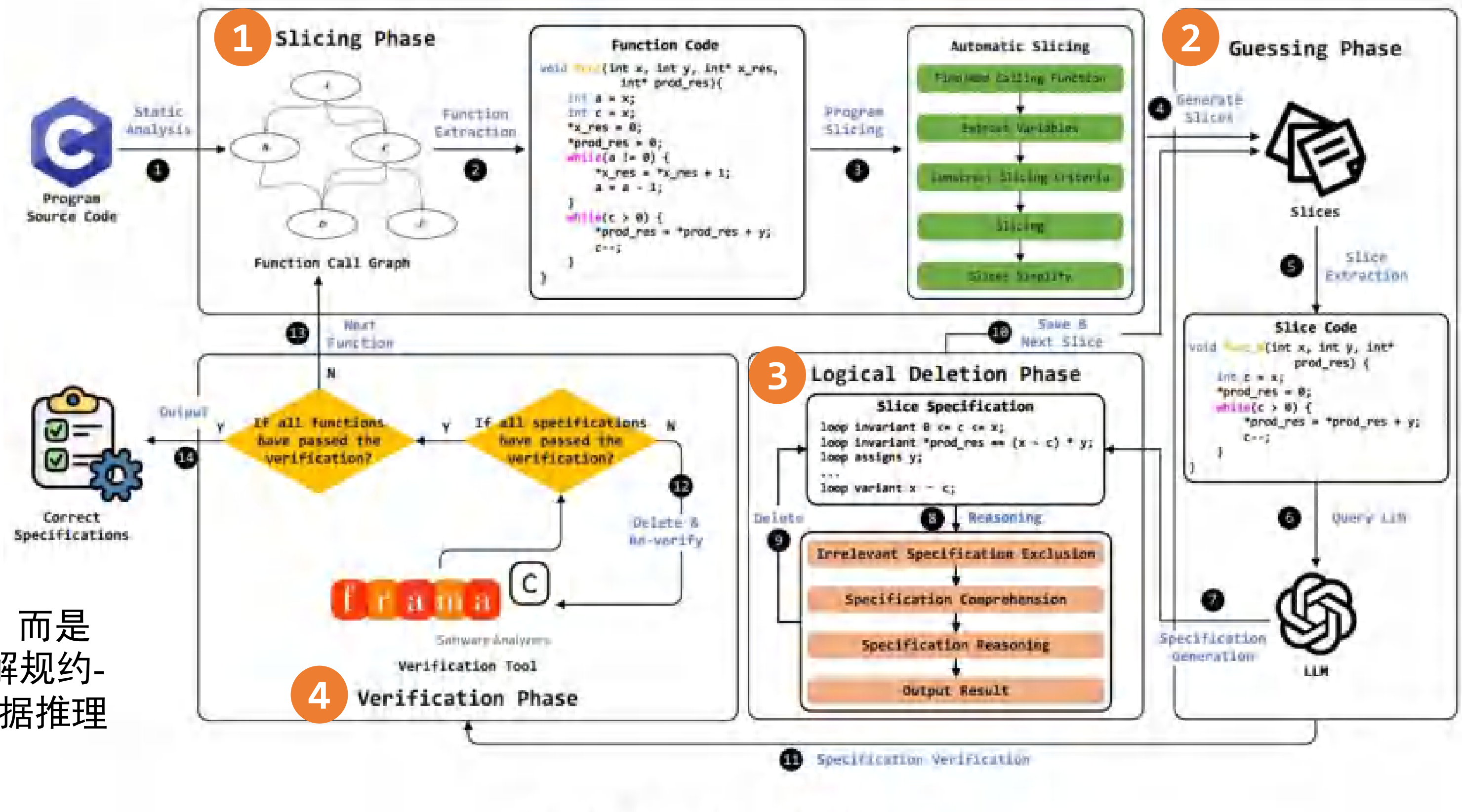


Fig. 2. Overview of SLD-Spec.

Slicing Phase：

通过静态分析构造函数调用图，选择其中一个函数，进行Slicing操作，得到该函数的若干个划分片。

Guess Phase：

对于每一个划分片，以few-shot的格式query大模型，生成规约。

Reasoning Phase：

获得规约后，不交予验证器，而是让大模型执行 ”排除无关规约-理解规约-评估规约是否合理” 流程，之后根据推理结果删除无关/错误规约。

Verification Phase：

触发调用验证器的条件是，当前函数内所有划分片已完成规约生成/所有函数已完成规约生成。

SLD-Spec:

Enhancement LLM-assisted Specification Generation for Complex Loop Functions via Program Slicing and Logical Deletion

Benchmark Information				AutoSpec		SLD-Spec	
Type	Program	LoC	NoL	NPP	RT (s)	NPP	RT (s)
general_wp_problems	absolute_value.c	13	0	5/5	54.00	5/5	12.46
	add.c	14	0	5/5	21.74	5/5	4.00
	ani.c	15	1	0/5	/	0/5	/
	diff.c	8	0	5/5	2.85	5/5	2.80
	gcd.c	17	0	0/5	/	0/5	/
	max_of_2.c	12	0	5/5	5.69	5/5	2.63
	power.c	17	1	0/5	/	0/5	/
	simple_interest.c	10	0	5/5	4.04	5/5	3.66
	swap.c	12	0	5/5	8.43	5/5	4.59
	triangle_angles.c	13	0	5/5	9.89	5/5	5.10
	triangle_sides.c	14	0	5/5	3.98	5/5	6.66
	wp1.c	13	0	0/5	/	0/5	/
pointers	add_pointers.c	15	0	5/5	25.01	5/5	3.52
	add_pointers_3_vars.c	16	0	5/5	12.13	5/5	3.82
	div_rem.c	11	0	5/5	3.90	5/5	11.88
	incr_a_by_b.c	12	0	5/5	5.71	5/5	7.39
	max_pointers.c	12	0	5/5	11.41	5/5	3.86
	order_3.c	31	0	0/5	/	0/5	/
	reset_1st.c	14	0	3/5	26.35	5/5	7.45
	swap.c	12	0	5/5	3.72	5/5	6.43
loops	1.c	8	1	5/5	6.35	5/5	14.58
	2.c	14	1	0/5	/	0/5	/
	3.c	15	1	0/5	/	4/5	19.93
	4.c	15	1	0/5	/	0/5	/
	fact.c	15	1	0/5	/	0/5	/
	mult.c	13	1	0/5	/	4/5	21.98
	sum_digits.c	15	1	0/5	/	0/5	/
	sum_even.c	14	1	0/5	/	0/5	/

Benchmark Information				AutoSpec		SLD-Spec	
Type	Program	LoC	NoL	NPP	RT (s)	NPP	RT (s)
immutable_arrays	array_sum.c	14	1	0/5	/	0/5	/
	binary_search.c	21	1	0/5	/	0/5	/
	check_evens_in_array.c	16	1	0/5	/	4/5	22.16
	max.c	22	1	5/5	45.04	5/5	38.28
	occurences_of_x.c	23	1	4/5	40.34	0/5	/
	sample.c	13	1	0/5	/	0/5	/
	search.c	15	1	0/5	/	3/5	22.91
	search_2.c	16	1	0/5	/	5/5	38.36
mutable_arrays	array_double.c	17	1	0/5	/	0/5	/
	bubble_sort.c	20	2	0/5	/	0/5	/
more_arrays	equal_arrays.c	14	1	2/5	19.30	2/5	24.60
	replace_evens.c	14	1	2/5	26.58	5/5	19.79
	reverse_array.c	17	1	0/5	/	0/5	/
arrays_and_loops	1.c	9	0	5/5	2.97	5/5	2.65
	2.c	19	1	5/5	47.31	5/5	42.01
	3.c	18	0	5/5	12.45	5/5	8.39
	4.c	15	1	0/5	/	5/5	21.83
	5.c	16	1	0/5	/	0/5	/
miscellaneous	array_find.c	17	1	2/5	22.82	2/5	23.84
	array_max_advanced.c	22	1	4/5	51.17	5/5	24.17
	array_swap.c	16	0	5/5	13.90	5/5	6.10
	increment_arr.c	16	1	0/5	/	0/5	/
	max_of_2.c	13	0	5/5	3.18	5/5	5.27
Overall				27	18.16	**32**	**13.85**

NoL: Number of Loop “循环个数”

NPP: Number of Program Verification Passes “整个程序验证通过的次数”

RT: Run Time “执行时间”

Table 4. Effectiveness of Different Schemes on the Complex-Loop Dataset.

Benchmark Information			AutoSpec+GPT-3.5-turbo			AutoSpec+DeepSeek-V2.5			SLD-Spec+GPT-3.5-turbo		
Program	LoC	NoL	PCRSAV	NAV	NPP	PCRSAV	NAV	NPP	PCRSAV	NAV	NPP
2-single-loop.c	20	2	60.74%(16.4)	0/2	0/5	72.39%(19.4)	1/2	0/5	100%(26.4)	2/2	5/5
3-single-loop.c	27	3	42.21%(20.6)	0/3	0/5	62.37%(24.2)	1/3	0/5	100%(38.8)	3/3	5/5
4-single-loop.c	34	4	32.84%(26.4)	0/4	0/5	55.97%(30)	0.8/4	0/5	100%(44.8)	4/4	5/5
Class Total			45.26%(21.13)	0/9	0%	63.58%(24.53)	2.8/9	0%	**100%(36.67)**	**9/9**	**100%**
only-single-loop-2.c	26	1	93.67%(14.8)	0/8	0/5	100%(12.4)	0/8	0/5	100%(19.4)	8/8	5/5
only-single-loop-3.c	31	1	95.45%(21)	0/9	0/5	100%(20.4)	0/9	0/5	100%(39.4)	9/9	5/5
only-single-loop-4.c	37	1	85.12%(20.6)	0/10	0/5	100%(22)	0/10	0/5	99.57%(45.8)	8.8/10	0/5
Class Total			91.41%(18.8)	0/27	0%	100%(18.27)	0/27	0%	**99.86%(34.87)**	**25.8/27**	**66.67%**
2-loop-1-if-m.c	35	2	38.25%(16.6)	0/3	0/5	79.01%(25.6)	2/3	0/5	100%(33.4)	3/3	5/5
2-loop-1-if-t.c	35	2	43.85%(16.4)	0/3	0/5	75.21%(18.2)	1/3	0/5	100%(31.4)	2.8/3	4/5
2-loop-1-if-b.c	35	2	51.58%(19.6)	1/3	0/5	75.17%(22.4)	2/3	0/5	100%(31.4)	2.8/3	4/5
Class Total			44.56%(17.53)	1/9	0%	76.46%(22.07)	5/9	0%	**100%(32.07)**	**8.6/9**	**86.67%**
nested-1.c	18	3	66.91%(18.2)	0/2	0/5	72.86%(10.2)	0/2	0/5	100%(35.2)	1.8/2	4/5
nested-2.c	18	3	63.95%(18.8)	0/2	0/5	75.74%(20.6)	0/2	0/5	100%(33.6)	1.4/2	3/5
Class Total			65.43%(18.5)	0/4	0%	74.3%(15.4)	0/4	0%	**100%(34.4)**	**3.2/4**	**70%**
Number of Passes			0			0			**10**		

PCRSAV: Proportion of Correct and Relevant Specifications After Verification
“验证后既正确又相关的规范占比”

NAV: Number of Assertions Passed Verification
“验证时有多少条断言被成功证明”

NPP: Number of Program Verification Passes
“整个程序验证通过的次数”