Codeflaws: A Programming Competition Benchmark for Evaluating Automated Program Repair Tools

Shin Hwei Tan*, Jooyong Yi#, Yulis*, Sergey Mechtaev*, Abhik Roychoudhury*
*National University of Singapore, #Innopolis University

Abstract

Several automated program repair techniques have been proposed to reduce the time and effort spent in bug-fixing. While these repair tools are designed to be generic such that they could address many software faults, different repair tools may fix certain types of faults more effectively than other tools. Therefore, it is important to compare more objectively the effectiveness of different repair tools on various fault types. However, existing benchmarks on automated program repairs do not allow thorough investigation of the relationship between fault types and the effectiveness of repair tools. We present *Codeflaws*, a set of 3902 defects from 7436 programs automatically classified across 39 defect classes (we refer to different types of fault as defect classes derived from the syntactic differences between a buggy program and a patched program).

Codeflaws

- 3092 defects extracted from CODEFORCES β Sponsored by Telegram
- Allows extensive investigation of repairable defect classes
- Contains scripts for running 4 state-of-the-art automated repair tools
  - GenProg, SPR, Prophet, Angelix

Our defect classes and example of each defect class

AST Type	Defect Type	Defect Class	Example
Statement	Control flow	(SDIF) Delete if, else, else if, for or while	- if (lines[i].y1 == last->y1)
		(SIIF) Insert if, else, else if, for or while	+ if(l)
		(SRIF) Replace if, else, else if, for or while	- if(a==b) + if(mask(a)==b)
		(SIRT) Insert return	+ return 0;
		(SDIB) Delete/Insert break or continue	- break;
	Data flow	(SDLA) Delete assignment	- answer+=((i-1)*dif);
		(SISA) Insert assignment	+ t=0;
	Function call	(SDFN) Delete function call	- printf("%s %s\n",s1,s2);
		(SISF) Insert function call	+ scanf("%d", &n);
	Type	(STYP) Replace variable declaration type	- int a; + long a;

	Move	(SMOV) Move statement	- scanf("%d",&i);
			scanf("%s", &a);
			+ scanf("%d",&i);

		(SMVB) Move brace up/down	- }
			printf("%d",c);
			+ }

Operator	Control flow	(ORRN) Replace relational operator	- if(sum>n) + if(sum>=n)
		(OLLN) Replace logical operator	- if((s[i] == '4') && (s[i] == '7')) + if((s[i] == '4') || (s[i] == '7'))
		(OILN) Tighten condition or loosen condition	- if(t%2==0) + if(t%2==0 && t!=2)
		(OEDE) Replace = with == or vice versa	- else if(n=1 && k==1) + else if(n==1 && k==1)
		(OICD) Insert a conditional operator	- printf ("%d\n", i); + printf ("%d\n", 3 == x ? 5 : i);
	Arithmetic	(OAAN) Replace arithmetic operator	- v2-=d; + v2+=d;
		(OAIS) Insert/Delete arithmetic operator	- max += days%2; + max += (days%7)%2;
		(OAID) Insert/Delete/Replace ++ or −−	+ i++;
		(OMOP) Modify operator precedence	- ans=max(ans,l-arr[n]*2); + ans=max(ans,(l-arr[n])*2);
	Function call	(OFFN) Alternative function call	- fflush(stdin); + getchar();
		(OFPF) Replace print format	- printf("%d\n",l); + printf("%lld\n",l);
		(OFPO) Modify function parameter order	- if(strcmp(c[i],b)>0) + if(strcmp(b,c[i])>0)
	Pointer	(OIRO) Insert/Delete Reference Operator	- printf("%d",&t); + printf("%d",t);
	Type	(OITC) Insert type cast operator	- ((p2m/p1m)*t+1 + ((float)p2m/p1m)*t+1;
OperanD	Constant	(DCCR) Replace constant with variable/constant	- for(i=n+1;i<=9000;i++) + for(i=n+1;i<=10000;i++)
	Variable	(DRVA) Replace a read variable with a variable/constant	- for (i=0;i<l;i++) + for (i=0;i<m;i++)
		(DRWV) Replace a write variable with a variable	- b=0; + a=0;
	Array	(DMAA) Insert/Replace array access	- out[l] = '\0'; + out[l−−] = '\0';
		(DRAC) Replace constant of array initialization	- int ex[2]={0,2}; + int ex[2]={0,3};
		(DCCA) Modify array size	- int x[100] + int x[100000];
Higher order	Non-branch	(HDMS) Delete multiple non-branch statements	- freopen("input.txt", "r", stdin); - freopen("output.txt", "w", stdout);

		(HIMS) Insert multiple non-branch statements	+ freopen("input.txt", "r", stdin);
			+ freopen("output.txt", "w", stdout);

		(HDIM) Delete and insert multiple non-branch statements	- break; + count=0;

	Branch stmt	(HBRN) Delete/Insert branch and non-branch statements	+ if(len%slov!=0){printf("NO");
			+ return 0;}

	Expressions	(HEXP) Delete/Insert/Replace operators & operands	- if(m*9>=s && s) + if((m*9>=s && !s) || (m==1))
	Combination	(HCOM) Insert/Replace statements and expressions	- rep(i,n) + for(i=n-1;i>=0;i−−)

	Others	(HOTH) Other higher order defect classes	- scanf("%s",h);
			+ for(i=0;i<71;i++)
			+ scanf("%c",&h[i]);

The Basic Statistics of Subject Programs in Codeflaws

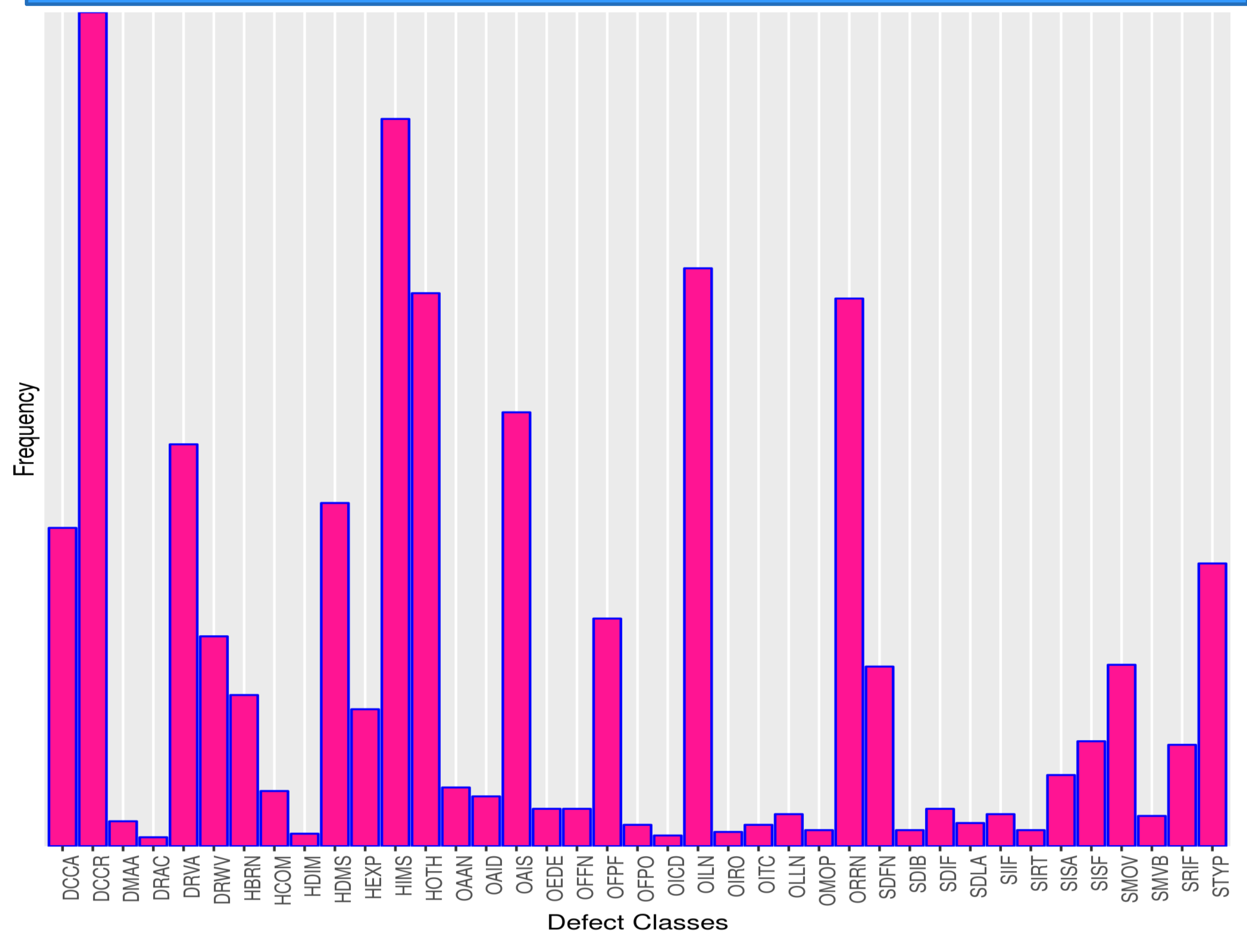
Measurement	Total/Range	Average
# of Programming Contest	548	-
# of Programming Problems	1284	-
# of Programs	7436	-
# of Defects	3902	-
Size of Repair Test Suite	2-8	3
Size of Held-out Test Suite	5-350	40
Source Lines of Codes	1-322	36

Our Criteria for Automated Program Repair Benchmark

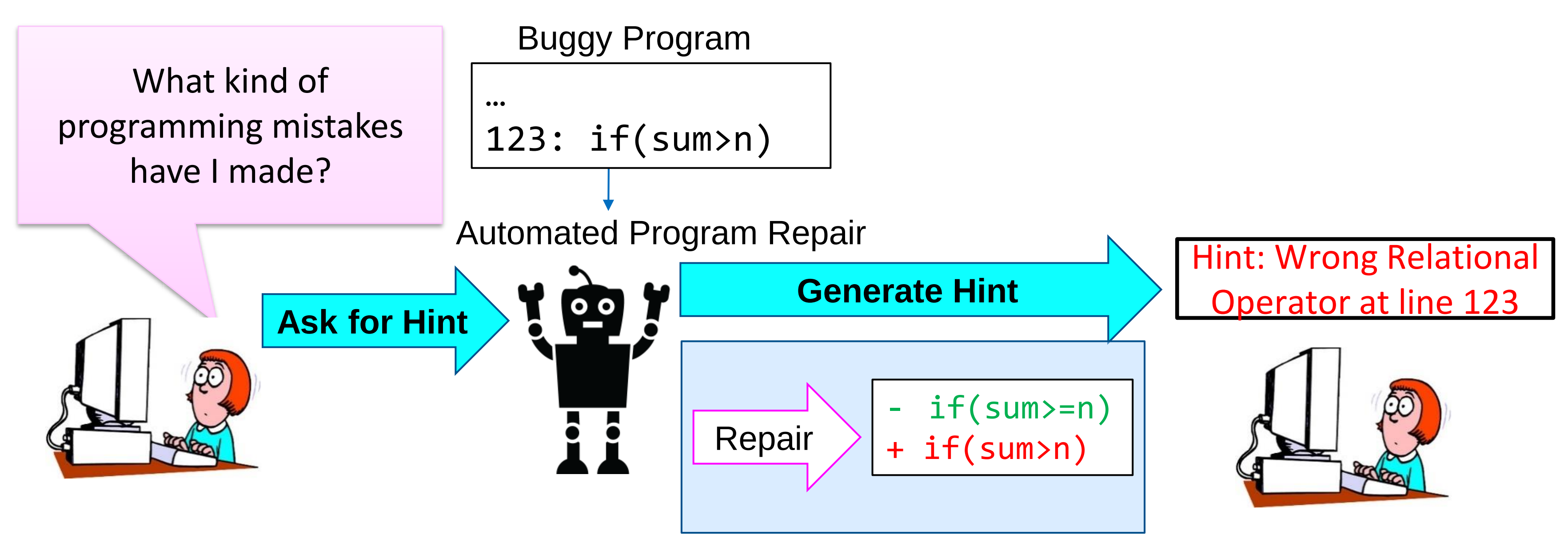
- ✓ **C1:** Diverse types of real defects.
- ✓ **C2:** Large number of defects.
- ✓ **C3:** Large number of programs.
- ✓ **C4:** Programs that are algorithmically complex
- ✓ **C5:** Large held-out test suite for patch correctness verification

- Defect class classification based on the syntactic differences between the buggy program and the patched program.
  1) Allows automatic classification of defect classes
  2) Enables extensive evaluation of different repair tools
  3) Commonly deployed in the literature

Distribution of defect classes



Example Usage In Intelligent Tutoring


What kind of programming mistakes have I made?
Ask for Hint
Buggy Program
Automated Program Repair
Generate Hint
Repair
Hint: Wrong Relational Operator at line 123

```
...
123: if(sum>n)
```

```
- if(sum>=n)
+ if(sum>n)
```

Conclusion

- Our *Codeflaws* benchmark aim to facilitate future empirical study in automated program repair.
- A step towards the evaluation of program repair tools against multiple dimensions with defect classes being one such dimension.
- *Publicly available for download at: https://codeflaws.github.io/*