13 B - -287113-1257 6-3821 877 354 constraints 332 6 977 877 78 877 -3471 6 8

$$
\begin{array}{ll}
\mathcal{C} \quad \{X \quad {}_{\mathsf{Set}}\, X\} \rightarrow \mathcal{C} & \mathcal{C} \quad \{c(\ldots) \quad {}_{\mathsf{Set}}\, d(\ldots)\} \rightarrow \text{inconsistent} \\
\mathcal{C} \quad \{e_{\mathsf{Set}} \quad {}_{\mathsf{Set}}\, 1\} \rightarrow \mathcal{C} & \mathcal{C} \quad \{c(\ldots) \quad {}_{\mathsf{Set}}\, 0\} \rightarrow \text{inconsistent} \\
\mathcal{C} \quad \{0 \quad {}_{\mathsf{Set}}\, e_{\mathsf{Set}}\} \rightarrow \mathcal{C} & \mathcal{C} \quad \{1 \quad {}_{\mathsf{Set}}\, c(\ldots)\} \rightarrow \text{inconsistent} \\
 & \mathcal{C} \quad \{
\end{array}
$$

			I		
X	Y_1	(1)	X	Z	(4)
X	Y_2	(2)	X	Z	(7)
Y_2	Z_1	(3)	Y_1	Z	(8)
Y_1	Z_1	(5)			
Z_1	Z_2	(6)			

Y_1 8

general, practical incremental algorithm for maintaining arbitrary data structures [7]. Adding ad-hoc support for incremental updates to each BANSHEE sort is daunting, as the algorithms are highly optimized. For example, our set con-

$$\frac{\Gamma(x) = \mathit{ref}(\ _x, X\ _x, \overline{X}\ _x)}{\Gamma\quad x : \mathit{ref}(\ _x, X\ _x, \overline{X}\ _x)}\ (\text{Var}) \qquad \frac{\Gamma\quad e : \tau}{\qquad\qquad \overline{\ }}\ (\text{Addr})$$

$$\frac{\Gamma\quad e : \tau\qquad \tau\quad \mathit{ref}(1, T, \overline{\ }}{\Gamma\quad *e : T}\ (\text{Deref}) \quad \frac{\begin{array}{c}\Gamma\quad e_1 : \tau_1 \qquad\qquad \Gamma\quad e_2 : \tau_2 \\ \tau_1\quad \mathit{ref}(1, 1, \overline{\ }\ _1)\qquad \tau_2\quad \mathit{ref}(1, T_2, \overline{0}) \\ T_2\quad T_1\end{array}}{\Gamma\quad e_1\ =\ e_2 : \tau_2}$$

1)

T

0)

(Assign)

Fig. 4. Constraint generation for Andersen's analysis

dynamic sets of constants are useful in many analyses. But all constants have a fixed, known signature, so generating them dynamically does not interfere with any of our static optimizations.

Each `data`

pointer operations. RCA approximates the set of classes to which each expression

improvements. Table 1 shows wall clock execution times in seconds for the benchmarks. Benchmark size is measured in preprocessed lines of code (the two largest benchmarks, gimp and Linux, are approximately 430,000 and 2.2 million source lines of code, respectively). We compile

11. D. Gay and A. Aiken. Language support for regions. In *IG*[illegible] *C* [illegible]
[illegible] *8( )585-42831-12313 -123133( )-25( )11( )-1[illegible]5( )-454 1([illegible]25( )-8(3[illegible]2831-12313 321*