분석기의 진행율 예측하기

이우석, 오학주, 이광근

서울대학교 프로그래밍 연구실

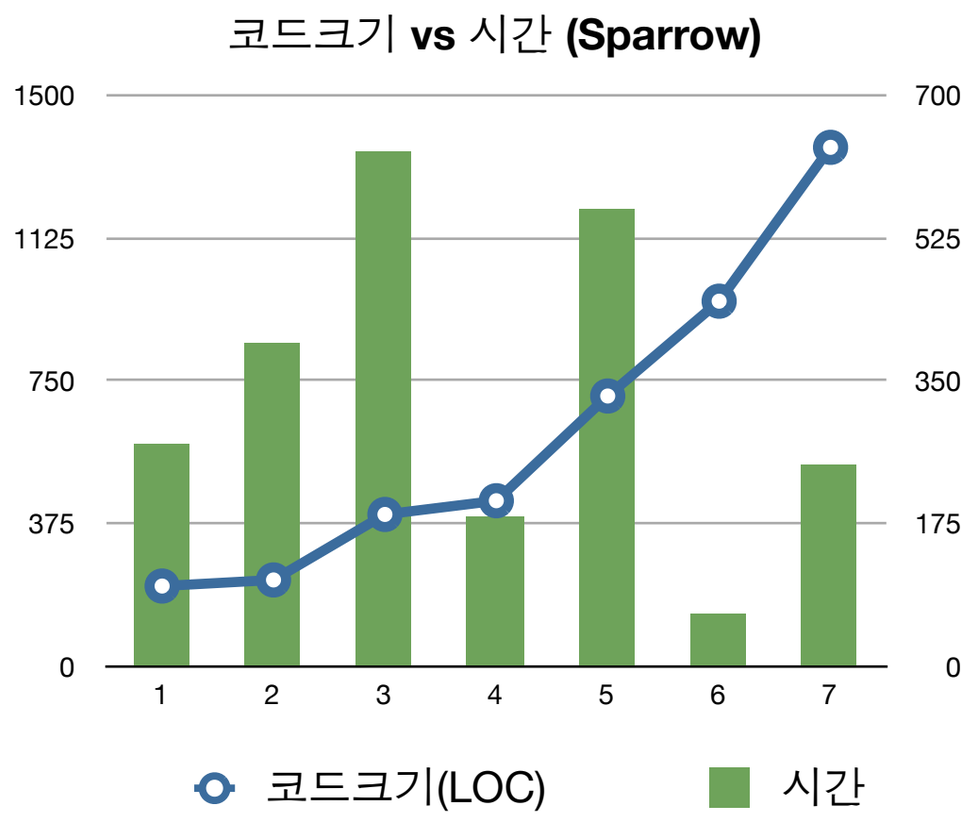
@ 11번째 ROSAEC 워크샵

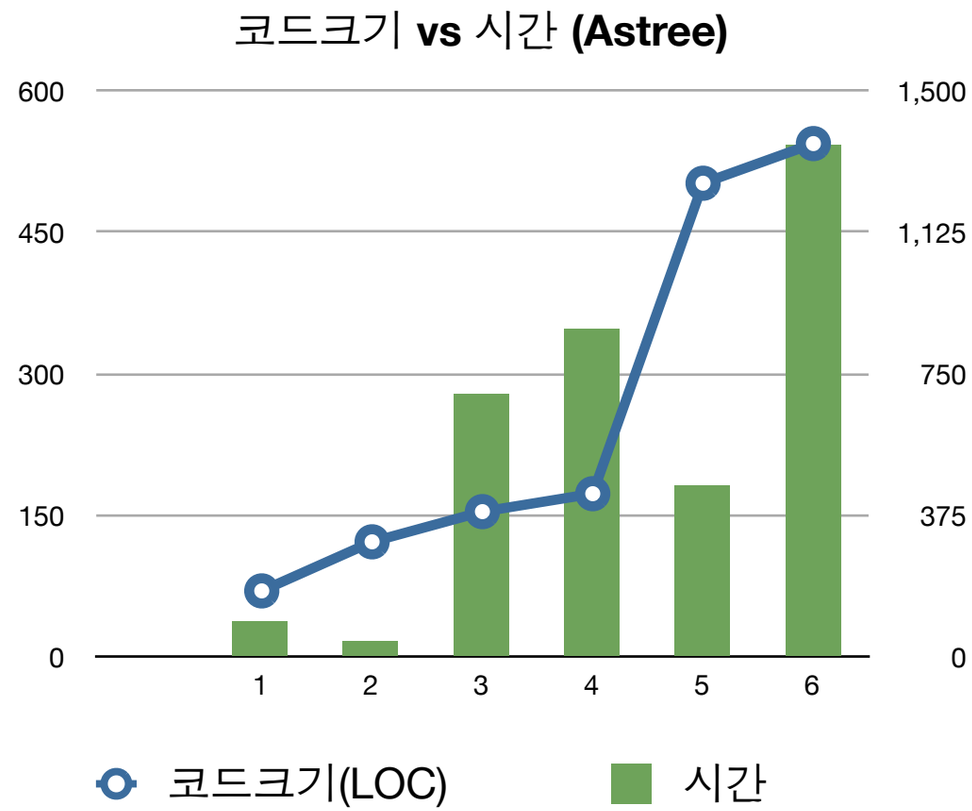
문제

- 크고 복잡해지는 소프트웨어, 오래걸리는 정적 분석 시간
  - Sparrow (SNU) - 40만줄 코드 10시간
  - Astrée (ENS) - 78만줄 코드 32시간
  - CGS (NASA) - 55만줄 코드 20시간
- 사용자가 진행율을 알 수 없는 문제

문제

- 코드 크기보다 의미적 복잡도 - 정확한 진행율을 알기 어려운 이유





우리의 방법

- 본 분석과 비슷한 사전 분석 + 기계학습
- 분석 진행율 예측에 대한 첫번째 연구
- 요약해석 기반 분석기에 일반적으로 적용가능

목표

- 정적 분석 : 요약 도메인이 $\mathbb{D}$, 요약 의미함수가 $F : \mathbb{D} \to \mathbb{D}$ 일때, 분석결과 $\mathrm{lfp}\, F$ 를 더 이상 변화가 없을 때까지 $F^n(\bot)$ 을 계산하여 얻음

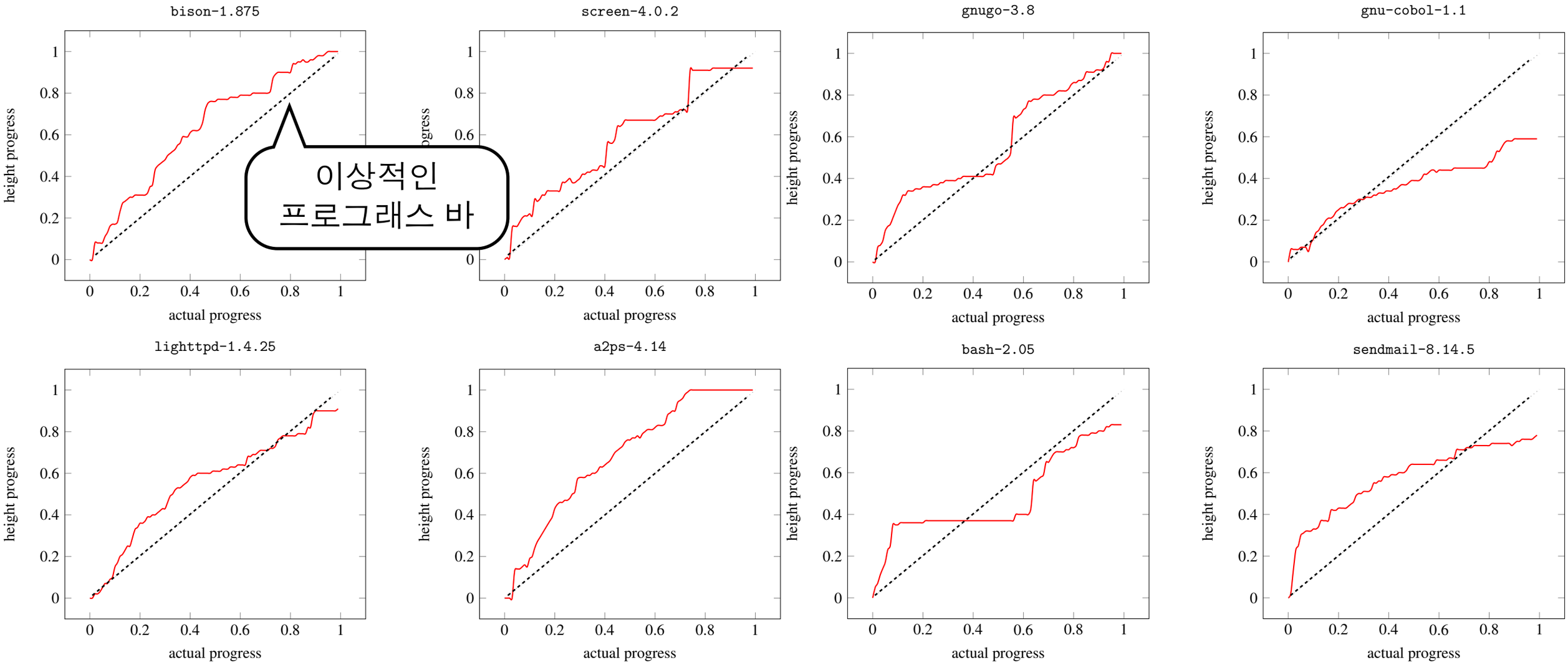
- 이상적인 프로그래스 바 : $\frac{i}{n} \quad (\mathrm{lfp}\, F = F^n(\bot))$

결과

- 실험 결과 : 8개 GNU 프로그램에 대해서 다음의 추가시간을 들여 진행율 예측
  - 인터벌 분석 **3.8%**
  - 포인터 분석 **7.3%**
  - 옥타곤 분석 **36.6%** (시험적 시도)

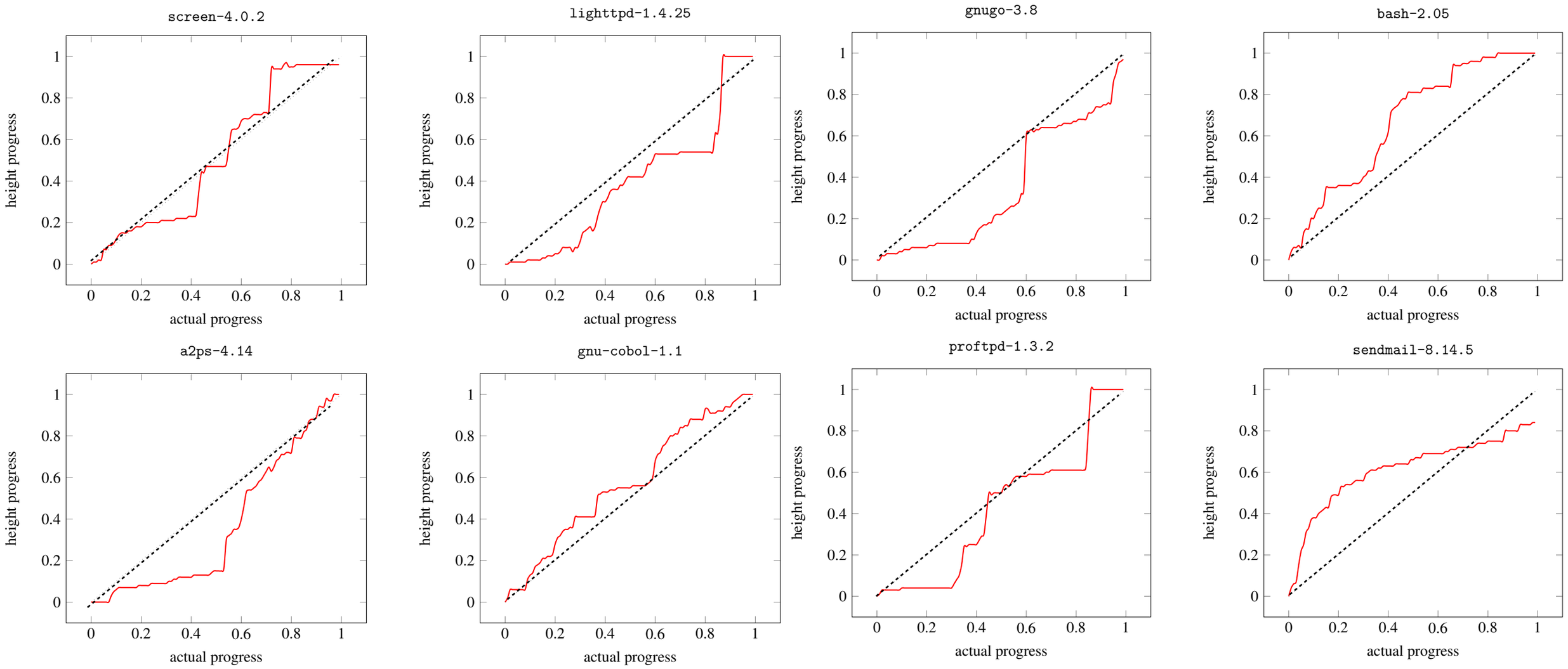
결과

- 인터벌 분석 (x축 : 실제 진행율 y축 : 추정 진행율)



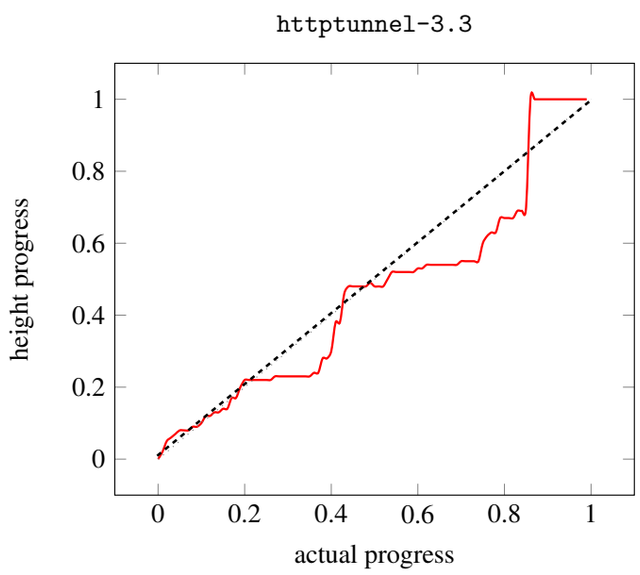
결과

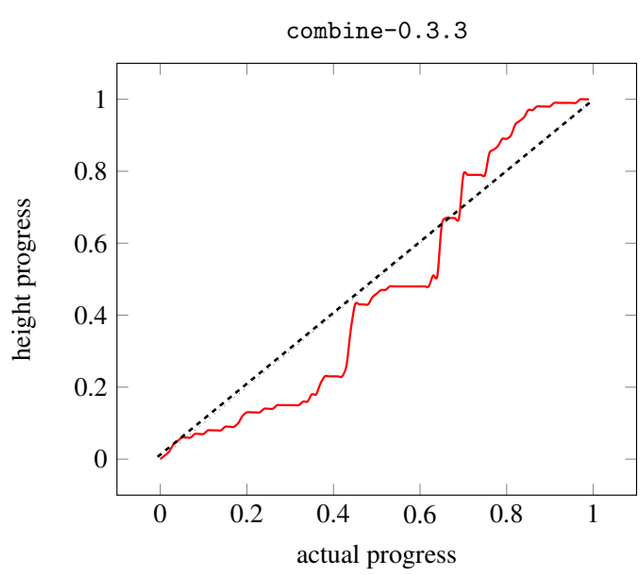
- 포인터 분석

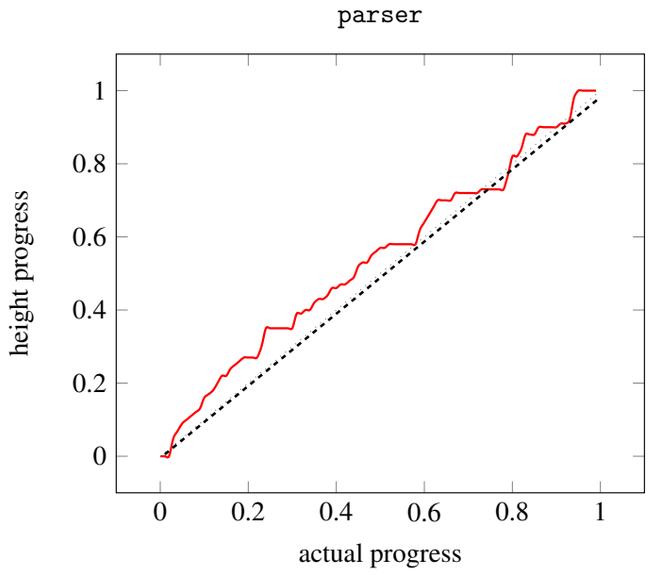

screen-4.0.2
lighttpd-1.4.25
gnugo-3.8
bash-2.05
a2ps-4.14
gnu-cobol-1.1
proftpd-1.3.2
sendmail-8.14.5
height progress
actual progress
0
0.2
0.4
0.6
0.8
1

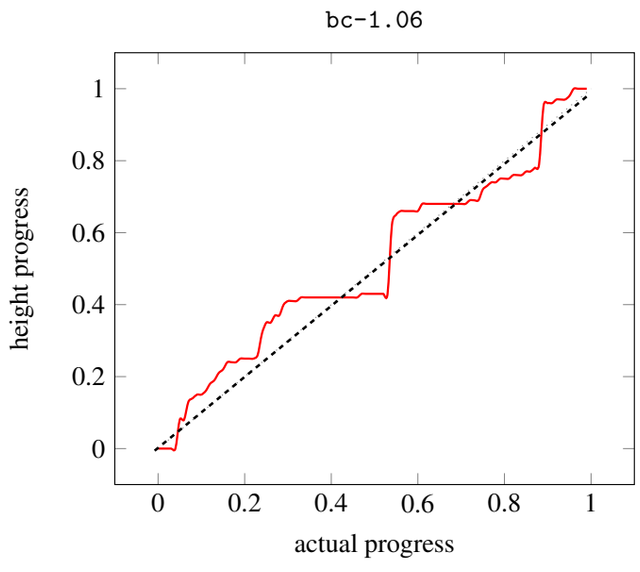
결과

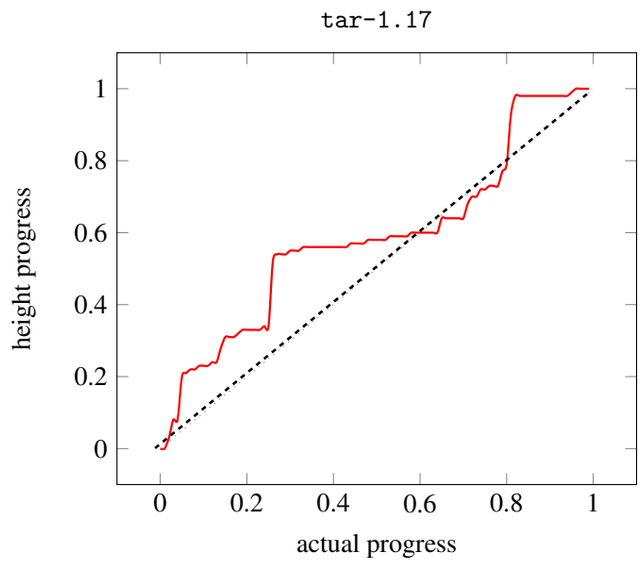
- 옥타곤 분석

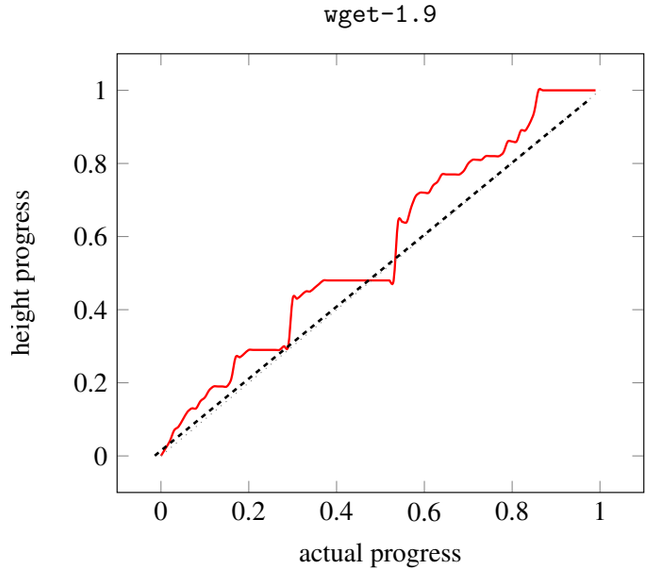

httptunnel-3.3
height progress
actual progress
0
0.2
0.4
0.6
0.8
1


combine-0.3.3
height progress
actual progress
0
0.2
0.4
0.6
0.8
1


parser
height progress
actual progress
0
0.2
0.4
0.6
0.8
1

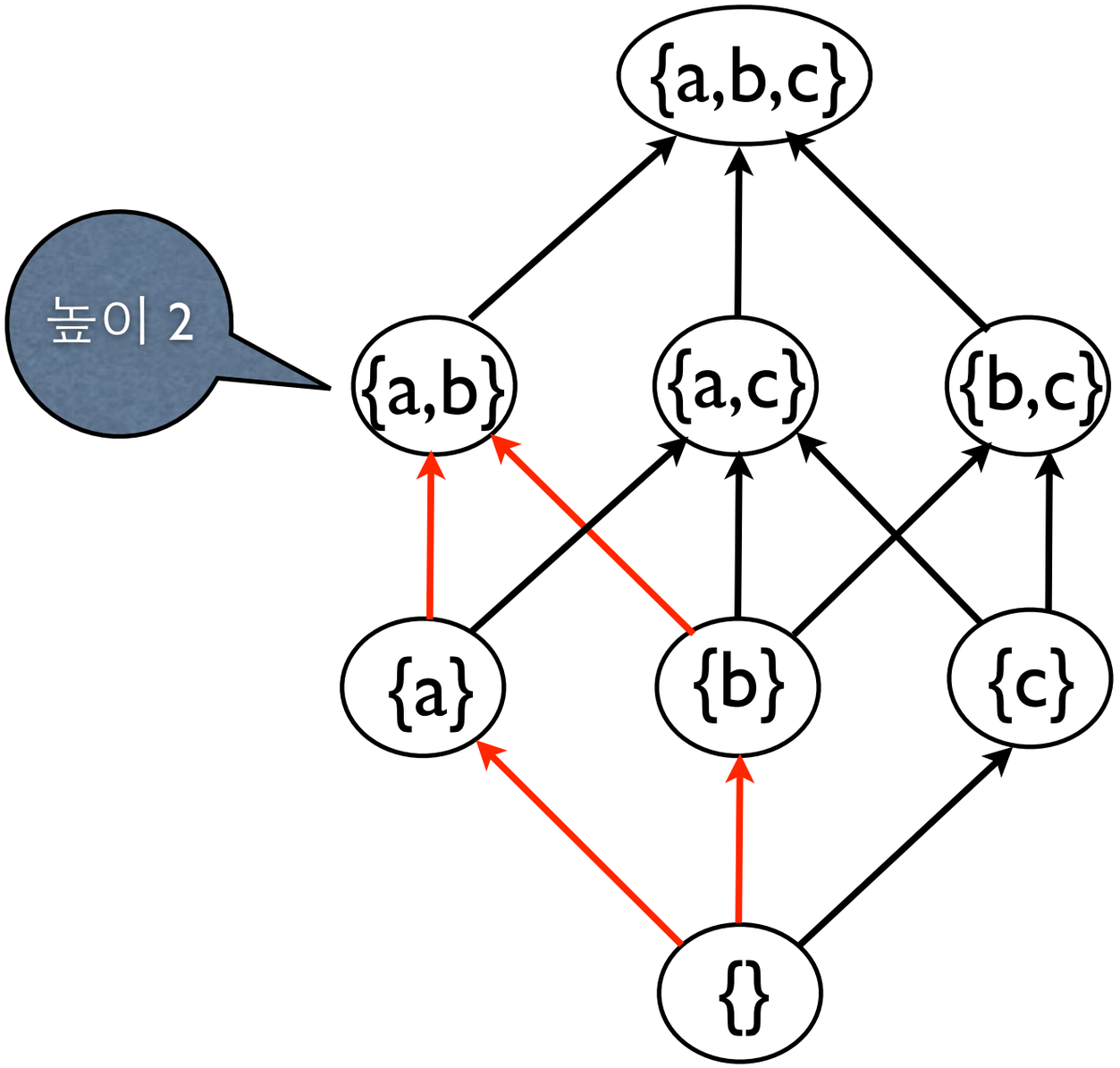

bc-1.06
height progress
actual progress
0
0.2
0.4
0.6
0.8
1


tar-1.17
height progress
actual progress
0
0.2
0.4
0.6
0.8
1

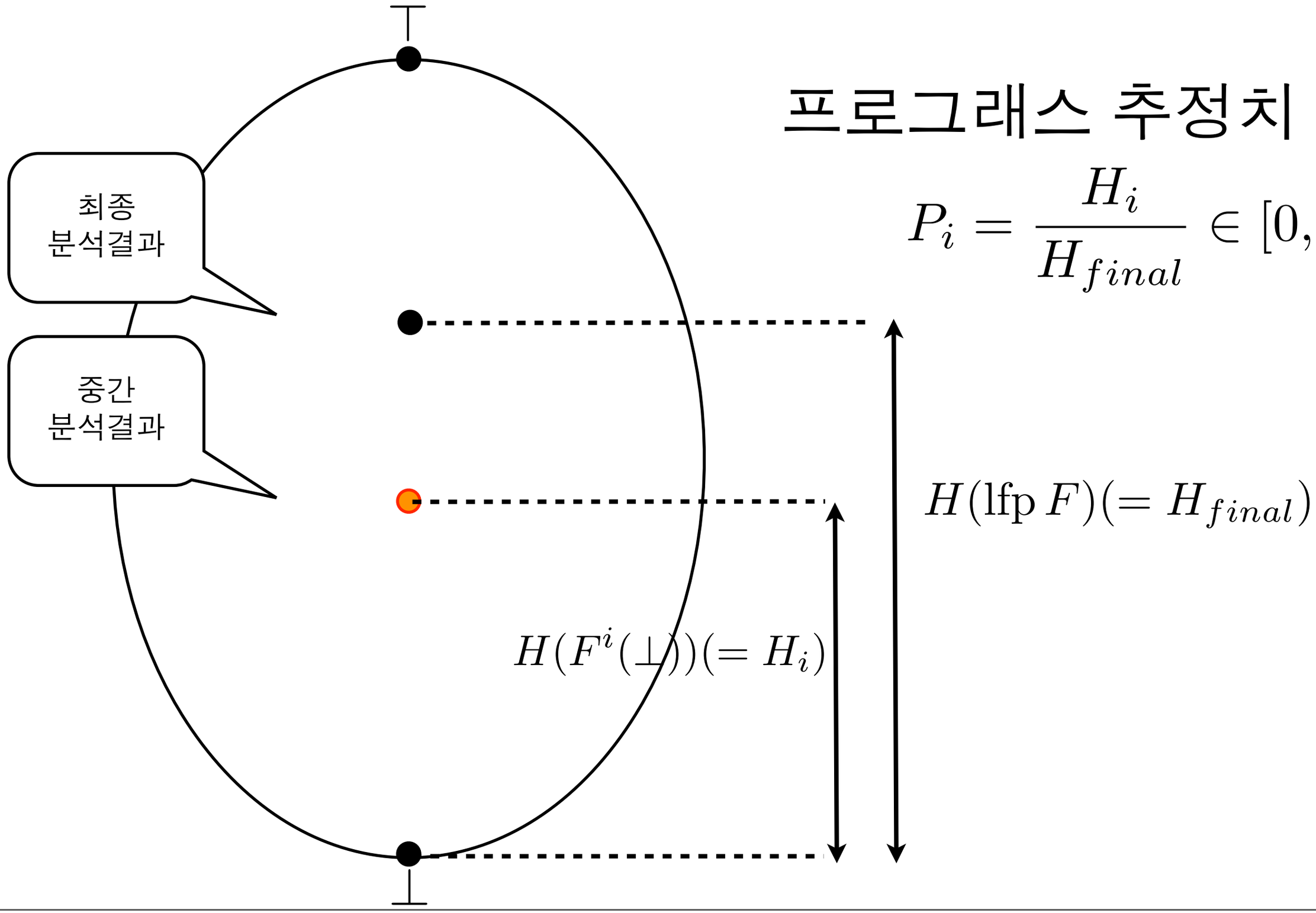

wget-1.9
height progress
actual progress
0
0.2
0.4
0.6
0.8
1

진행율 예측 방법

래티스와 원소의 높이



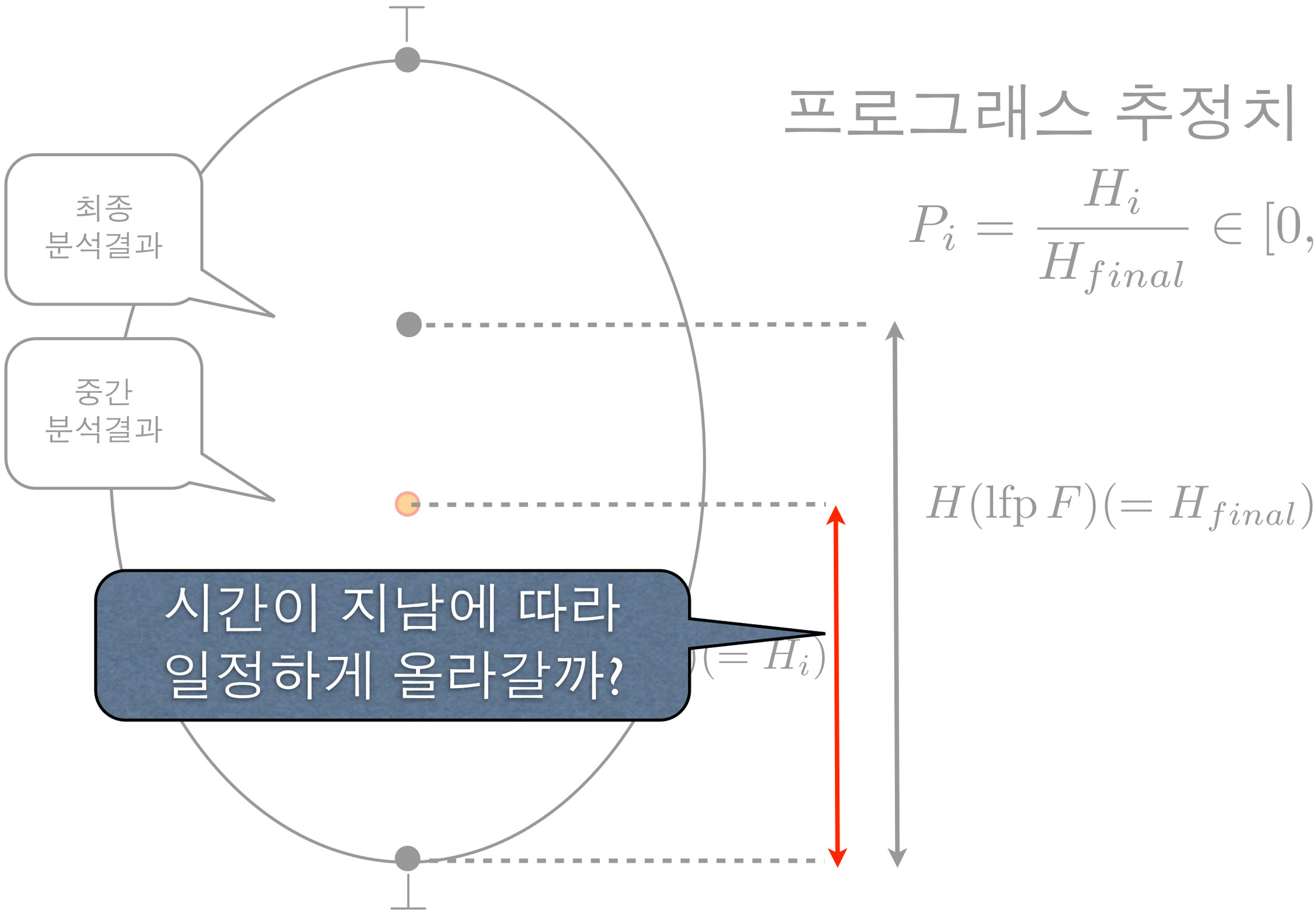
우리의 방법



프로그래스 추정치 :

$$P_i = \frac{H_i}{H_{final}} \in [0, 1]$$

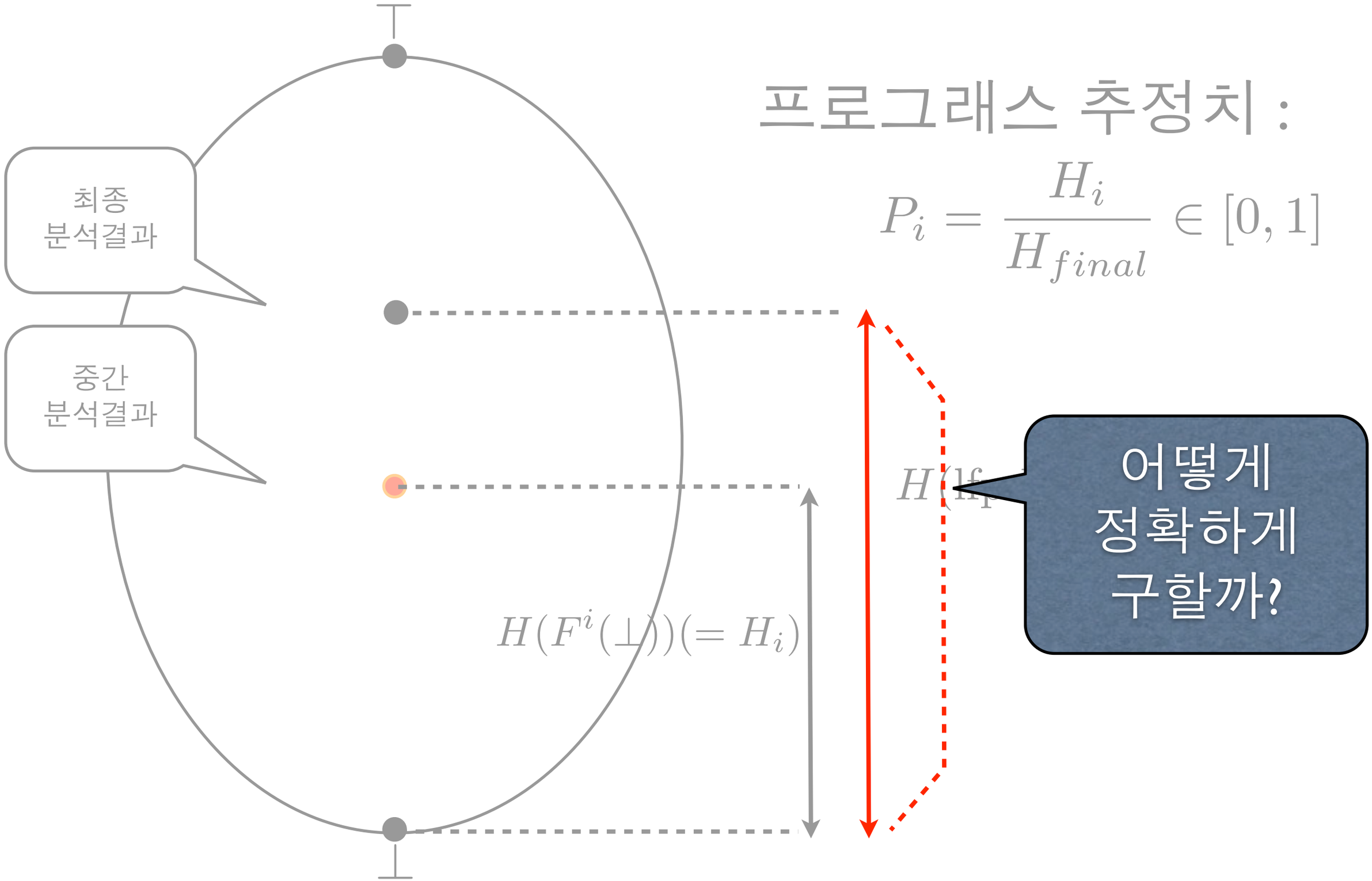
문제 1



프로그래스 추정치 :

$$P_i = \frac{H_i}{H_{final}} \in [0, 1]$$

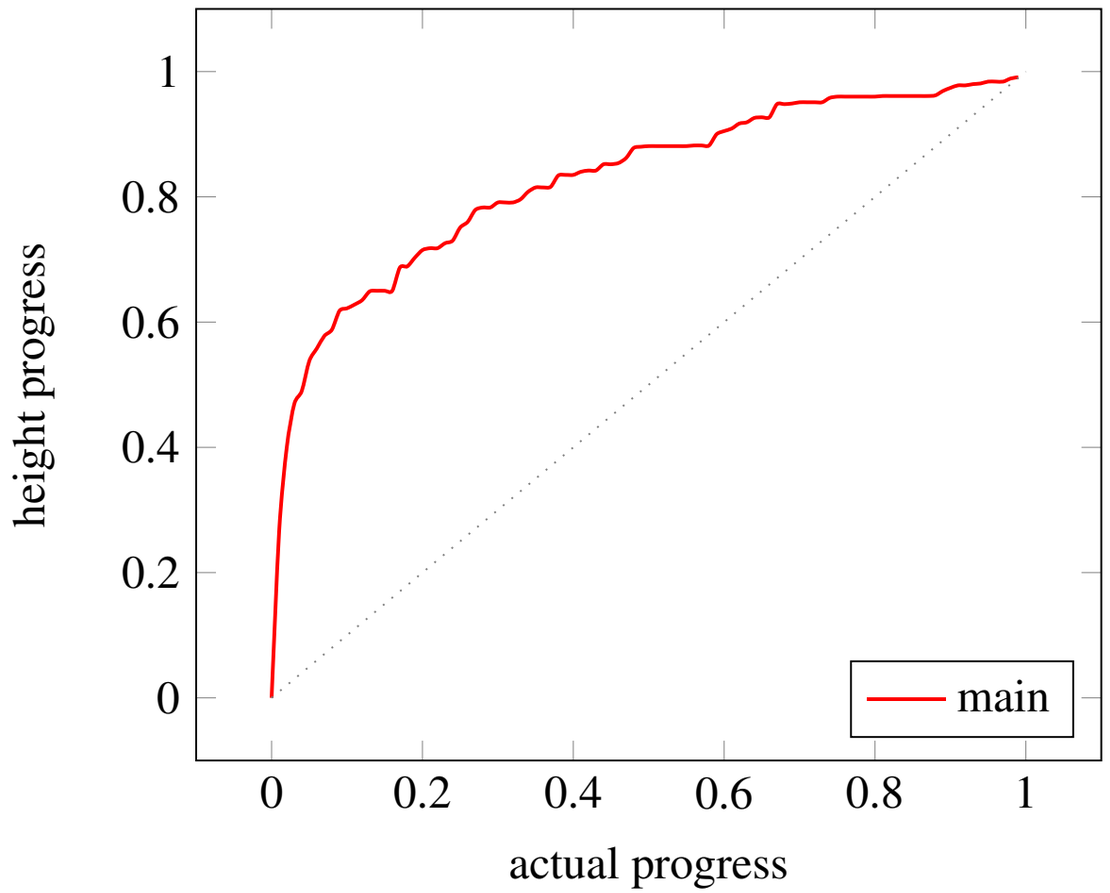
문제 2



프로그래스 추정치 :

$$P_i = \frac{H_i}{H_{final}} \in [0, 1]$$

분석 진행율



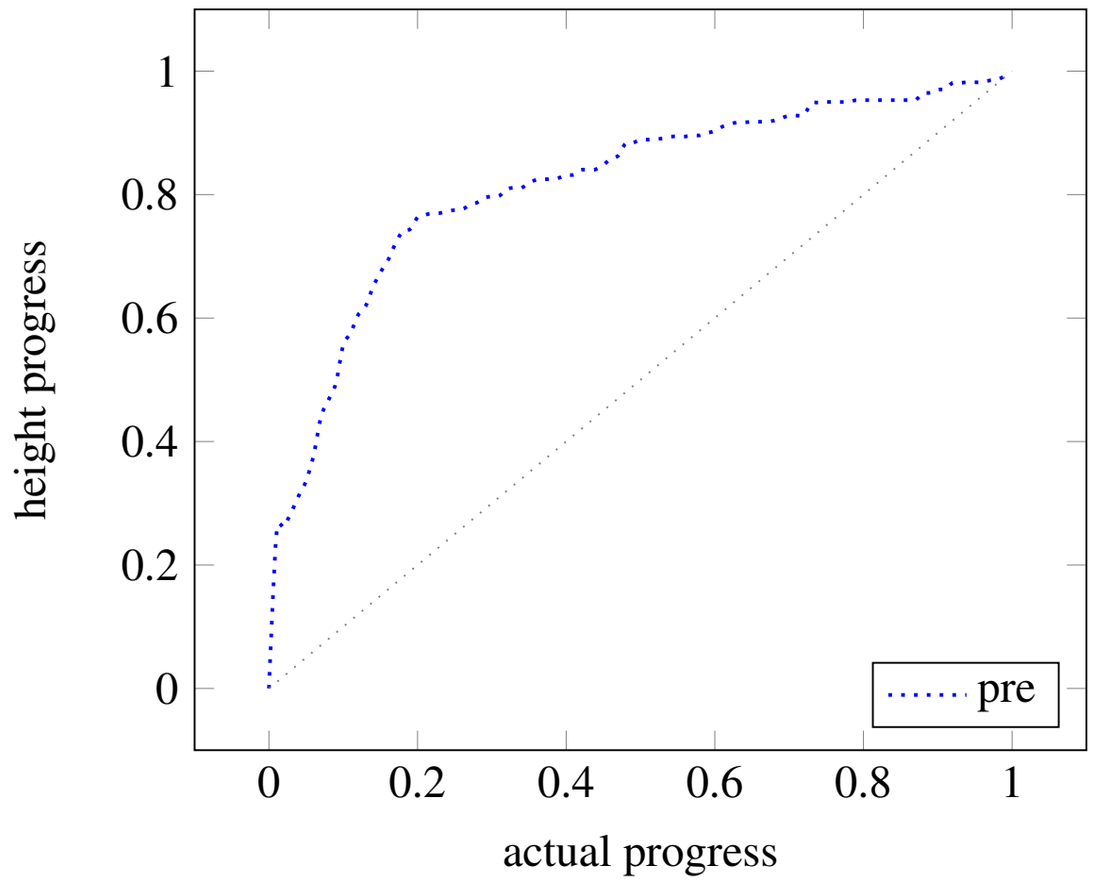
sendmail-8.14.6 (interval analysis)

해법

- 다음의 전분석(pre-analysis) 설계
- 본 분석보다 부정확(= 본분석 결과를 포섭함)
- 비용이 훨씬 저렴

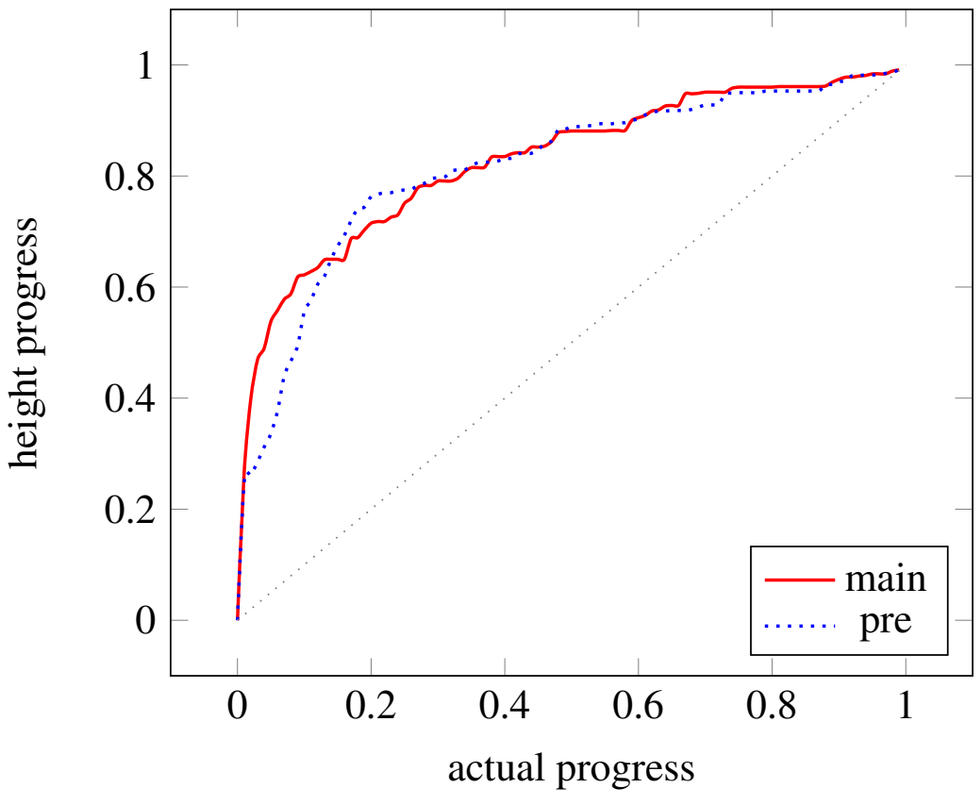
전분석 진행율

본 분석시간의 6.6%의 분석시간으로 수행

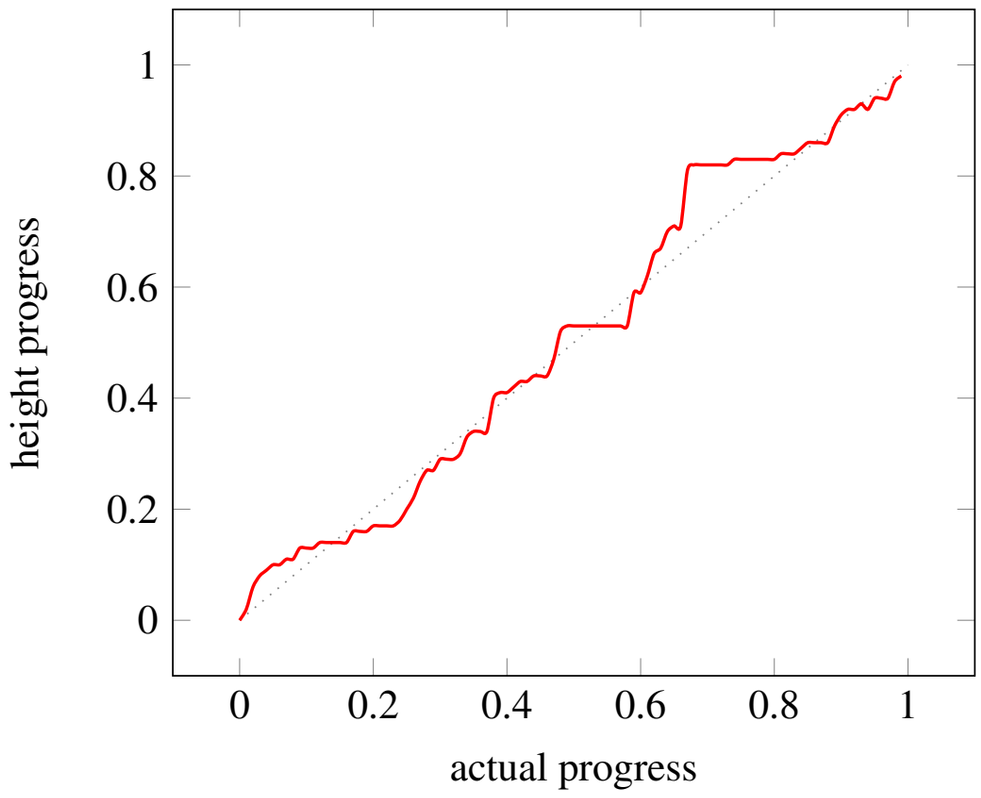


sendmail-8.14.6 (interval analysis)

전분석 진행율을 이용한 보정



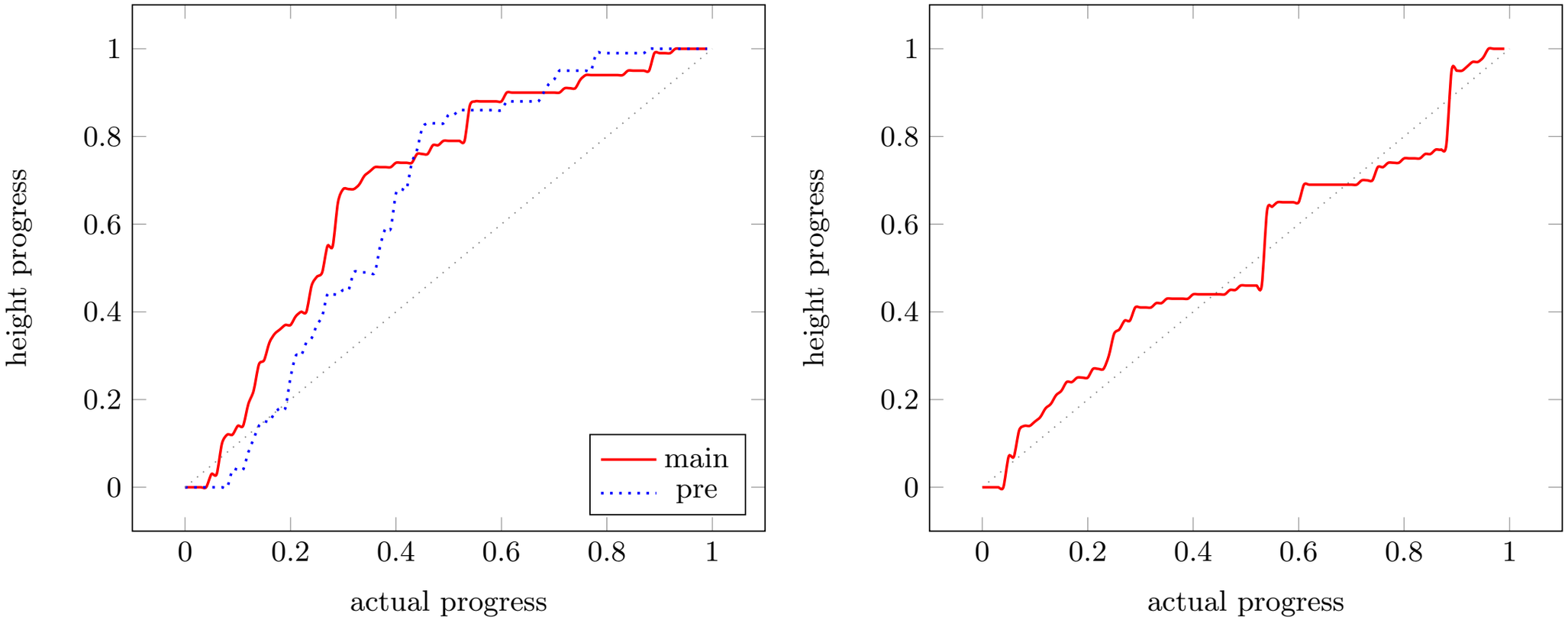
(a) original height-progress



(b) normalized height-progress

sendmail-8.14.6 (interval analysis)

전분석 진행율을 이용한 보정



(a) original height-progress

(b) normalized height-progress

wget-1.9 (octagon analysis)

진행율 보정

- I단계 : 본 분석보다 더 요약된(부정확한) 사전분석을 설계

$$\mathbb{D} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{D}^\sharp$$

$$F^\sharp : \mathbb{D}^\sharp \to \mathbb{D}^\sharp \qquad\qquad \alpha \circ F \sqsubseteq F^\sharp \circ \alpha$$

$$\bigsqcup_{i \in \mathbb{N}} F^{\sharp i}(\bot^\sharp) = F^{\sharp 0}(\bot^\sharp) \sqcup F^{\sharp 1}(\bot^\sharp) \sqcup F^{\sharp 2}(\bot^\sharp) \sqcup \cdots$$

진행율 보정

- 2단계 : 다음 데이터를 사전분석 실행 중 기록
  (사전분석은 m번의 iteration으로 고정점에 도달한다고 가정)

$$(\frac{H_0^\sharp}{H_m^\sharp}, \frac{0}{m}), \quad (\frac{H_1^\sharp}{H_m^\sharp}, \frac{1}{m}), \quad \cdots, \quad (\frac{H_i^\sharp}{H_m^\sharp}, \frac{i}{m}), \quad \cdots, \quad (\frac{H_m^\sharp}{H_m^\sharp}, \frac{m}{m})$$

where $H_i^\sharp = \mathsf{H}(\gamma(F^{\sharp i}(\bot^\sharp)))$.

진행율 보정

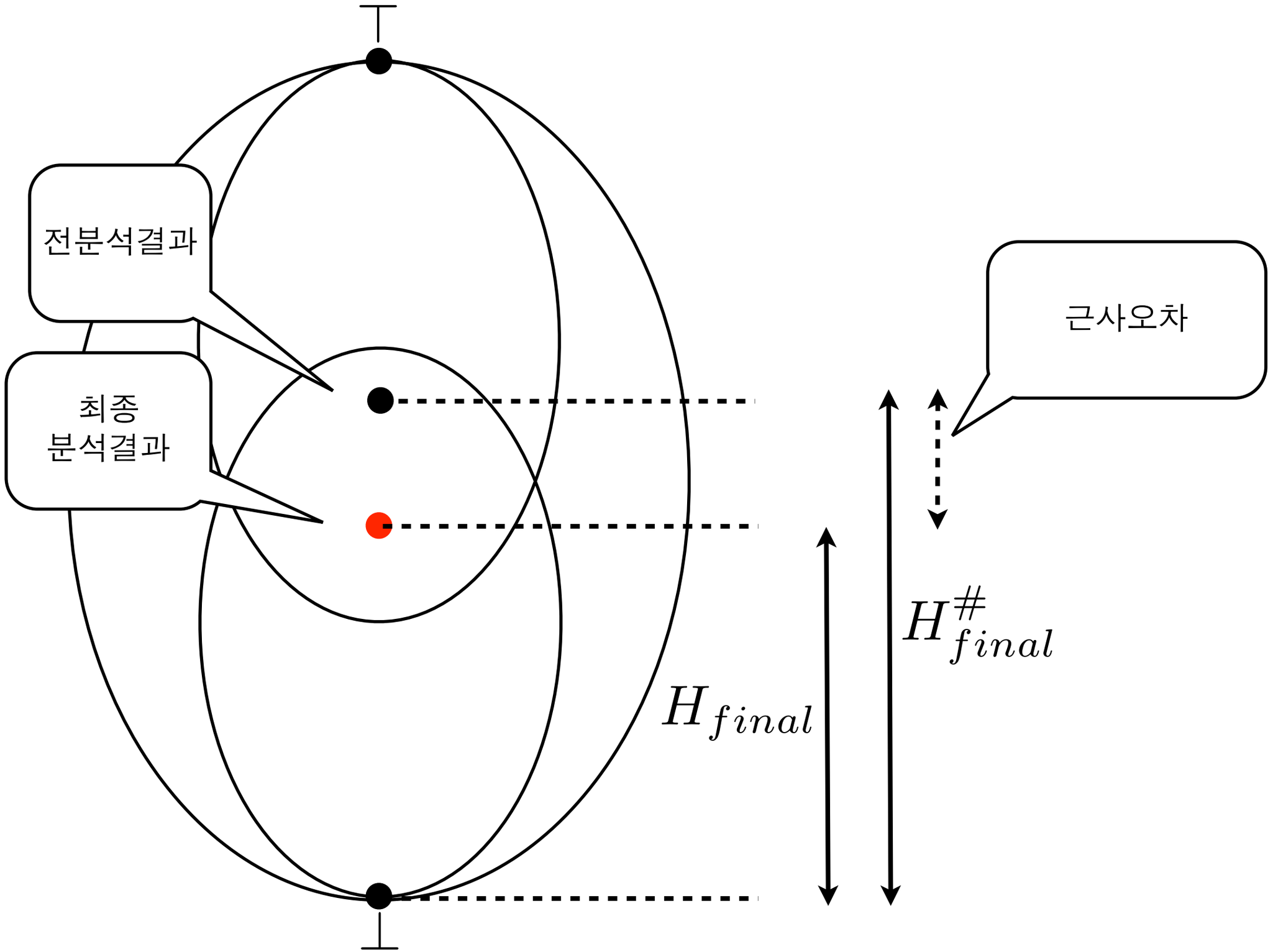
- 3단계 : 기록된 값을 선형보간하여 다음 함수를 얻음. 본 분석 실행 중 아래함수를 이용하여 프로그래스 보정

$$\text{normalize} : [0,1] \rightarrow [0,1]$$

우리의 가정

- 전분석이 본분석과 의미적으로 관계가 있다면
- 전분석의 래티스 높이비율 증가양상은 본분석의 그것과 닮았을 것이다.
- 실험적으로 우리의 가정을 증명

최종 높이 어림잡기



기계학습을 이용한 보정

- 다음 α 를 학습하여 정확한 최종 높이 계산

$$H_{final} = \alpha \cdot H^{\#}_{final} (0 \leq \alpha \leq 1)$$

기계학습을 이용한 보정

- 프로그램 구조, 전분석 결과와 관련된 값들을 이용하여 Ridge 선형 회귀 분석

Table 1. The feature vector used by linear regression to construct prediction models

Category	Feature
Inter-procedural (syntactic)	# function calls in the program
	# functions in recursive call cycles
	# undefined library function calls
Loop-related (syntactic)	the maximum loop size
	the average loop sizes
	the standard deviation of loop sizes
	the standard deviation of depths of loops
	# loopheads
Numerical analysis (semantic)	# bounded intervals in the pre-analysis result
	# unbounded intervals in the pre-analysis result
Pointer analysis (semantic)	# points-to sets of cardinality over 4 in the pre-analysis result
	# points-to sets of cardinality under 4 in the pre-analysis result
Post-fixpoint (semantic)	# program points where applying the transfer function once improves the precision
	height decrease when transfer function is applied once

기계학습을 이용한 보정

- 254개의 프로그램을 이용하여 학습
- 정확도 (3-fold validation) (대상값 $\alpha \in [0, 1]$)
  - 인터벌 : 평균오차 0.06, 표준편차 0.007
  - 포인터 : 평균오차 0.05, 표준편차 0.001
- 정확도 (랜덤 9개 프로그램 선별)
  - 평균오차 : 인터벌 - 0.07, 포인터 - 0.03

세부사항들

진행율 예측 세부사항들

- 우리가 고려하는 정적 분석
- 래티스 높이 함수
- 사전분석 디자인
- 최종 래티스높이 정확하게 어림잡기

진행율 예측 세부사항들

- 우리가 고려하는 정적 분석
- 래티스 높이 함수
- 사전분석 디자인
- 최종 래티스높이 정확하게 어림잡기

정적분석

- 프로그램 : $\langle \mathbb{C}, \hookrightarrow \rangle$
- 요약도메인 : 프로그램 지점에서 요약상태로의 맵핑:

$$\mathbb{D} = \mathbb{C} \to \mathbb{S}$$

- 요약 상태 : 변수에서 요약 값으로의 맵핑

$$\mathbb{S} = \mathbb{L} \to \mathbb{V}$$

- 요약 의미함수: $F \in (\mathbb{C} \to \mathbb{S}) \to (\mathbb{C} \to \mathbb{S})$

$$F(X) = \lambda c \in \mathbb{C}. f_c(\bigsqcup_{c' \hookrightarrow c} X(c'))$$

where $f_c \in \mathbb{S} \to \mathbb{S}$ is the transfer function for control point c.

진행율 예측 세부사항들

- 우리가 고려하는 정적 분석
- 래티스 높이 함수
- 사전분석 디자인
- 최종 래티스높이 정확하게 어림잡기

높이 함수

$$\mathsf{H} : (\mathbb{C} \to \mathbb{S}) \to \mathbb{N}$$

$$\mathsf{H}(X) = \sum_{c \in \mathbb{C}} \sum_{l \in \mathbb{L}} \mathsf{h}(X(c)(l))$$

- 인터벌 $\mathsf{h}(\bot) = 0$

$$\mathsf{h}([a,b]) = \begin{cases} 1 & a = b \ \wedge \ a, b \in \mathbb{Z} \\ 2 & a < b \ \wedge \ a, b \in \mathbb{Z} \\ 3 & a \in \mathbb{Z} \ \wedge \ b = +\infty \\ 3 & a = -\infty \ \wedge \ b \in \mathbb{Z} \\ 4 & a = -\infty \ \wedge \ b = +\infty \end{cases}$$

- 포인터 $\mathsf{h}(S) = \begin{cases} 4 & |S| \geq 4 \\ |S| & otherwise \end{cases}$
- 옥타곤
  - 옥타곤이 모든 임의의 변수 x,y에 대해 x+y, x-y 의 인터벌 범위를 나타내므로 인터벌 높이 함수를 그대로 사용.

진행율 예측 세부사항들

- 우리가 고려하는 정적 분석
- 래티스 높이 함수
- 사전분석 디자인
- 최종 래티스높이 정확하게 어림잡기

사전분석

- 본분석은 모든 프로그램 지점에서 요약상태 계산
- 사전분석은 특정 프로그램 지점들을 제외한 나머지 지점들은 요약상태를 뭉뚱그림
- 특정 프로그램 지점 : 반복문 입구 및 근처 지점들

부분적 흐름민감 분석

- 요약 도메인

$$\mathbb{C} \to \mathbb{S} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Delta \to \mathbb{S} \qquad \Delta = \Phi \cup \{\bullet\} \qquad \gamma(X) = \lambda c.\ X(\delta(c))$$

- 흐름을 구분짓는 프로그램 지점들 Φ ($\mathbb{W}$: 루프 입구)

$$\Phi = \{c \in \mathbb{C} \mid w \in \mathbb{W} \ \wedge \ c \hookrightarrow^{depth} w\}$$

조정가능한
매개변수 ∈[0,∞]

- 요약 의미함수

$$F^\sharp(X) = \lambda i \in \Delta.\ (\bigsqcup_{c \in \delta^{-1}(i)} f_c(\bigsqcup_{c' \hookrightarrow c} X(\delta(c'))))$$

$$\delta^{-1}(i) = \{c \in \mathbb{C} \mid \delta(c) = i\}$$

$$\delta(c) = \begin{cases} c & c \in \Phi \\ \bullet & c \notin \Phi \end{cases}$$

진행율 예측 세부사항들

- 우리가 고려하는 정적 분석
- 래티스 높이 함수
- 사전분석 디자인
- 최종 래티스높이 정확하게 어림잡기

최종 래티스 높이 근사

- 우리는 최종 래티스 높이의 근사치를 사전분석결과로부터 얻음

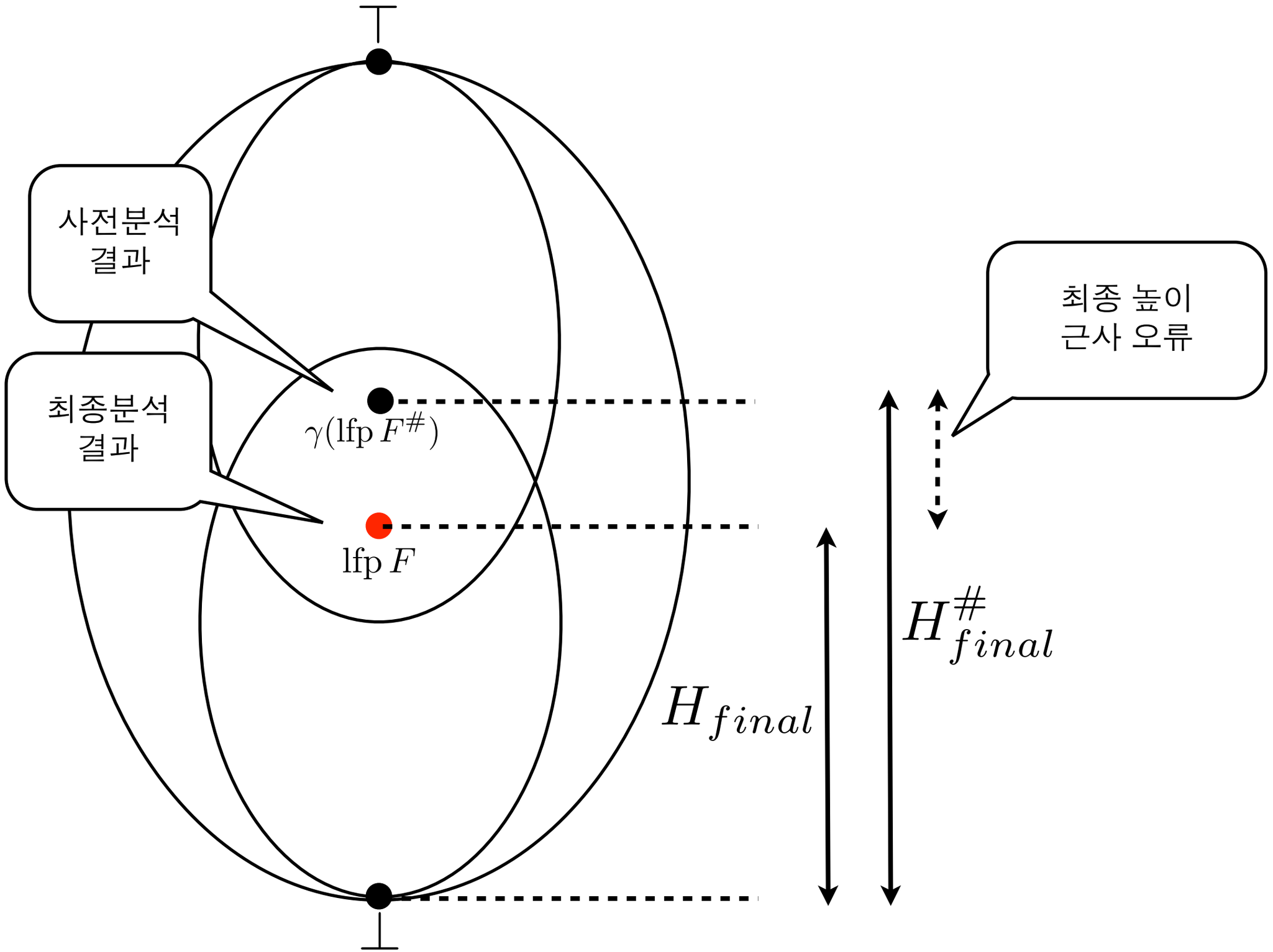
$$H^\sharp_{final} = \mathsf{H}(\gamma(\mathbf{lfp}F^\sharp))$$

- 사전분석은 본분석결과를 포괄하므로 :

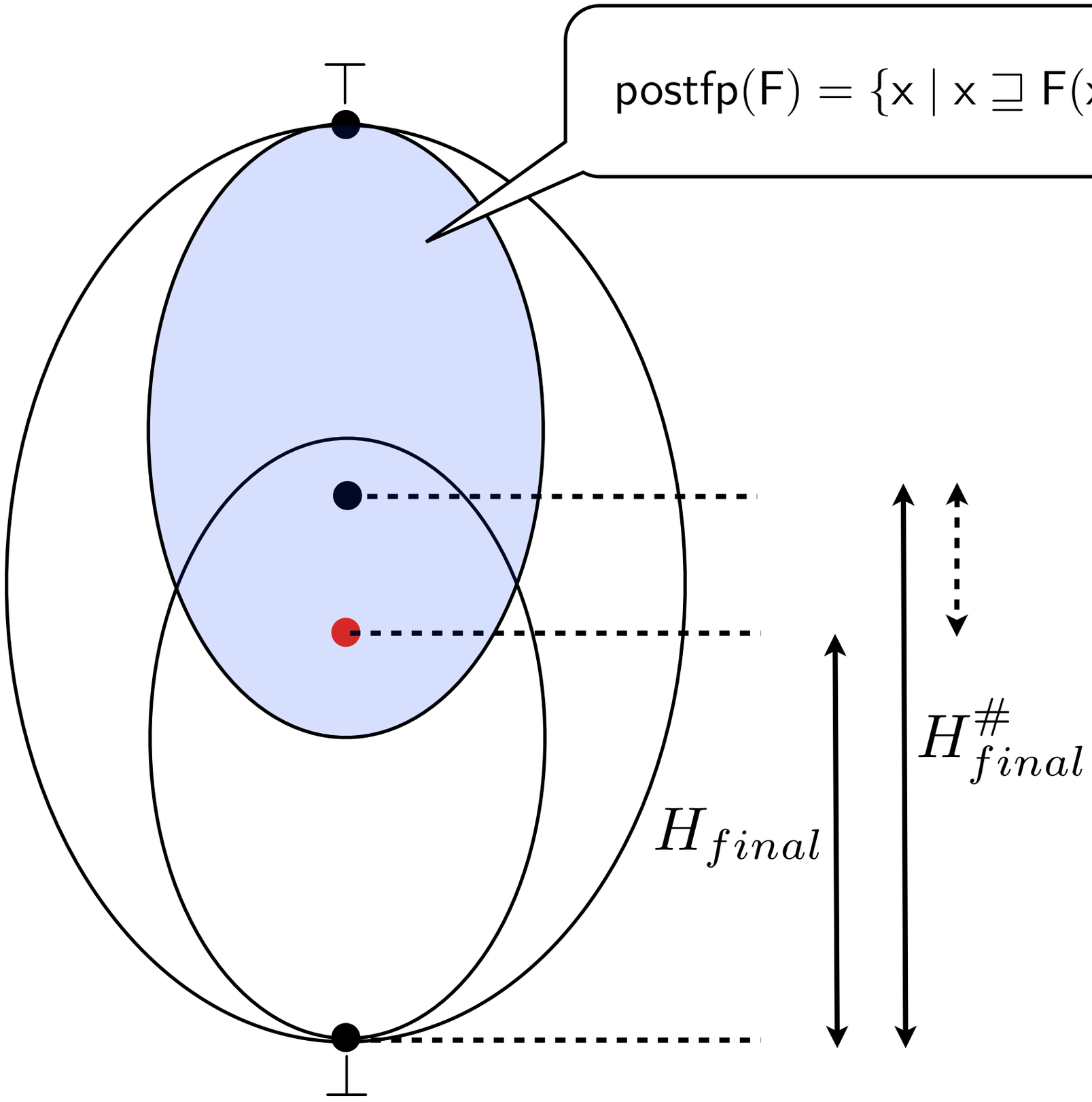
$$H_{final} = \mathsf{H}(\mathbf{lfp}F) \leq \mathsf{H}(\gamma(\mathbf{lfp}F^\sharp)) = H^\sharp_{final}$$

- 최종 높이 근사 오류$(H^\sharp_{final} - H_{final})$ 가 클 경우 분석 진행율 예측값이 망가질 수 있음

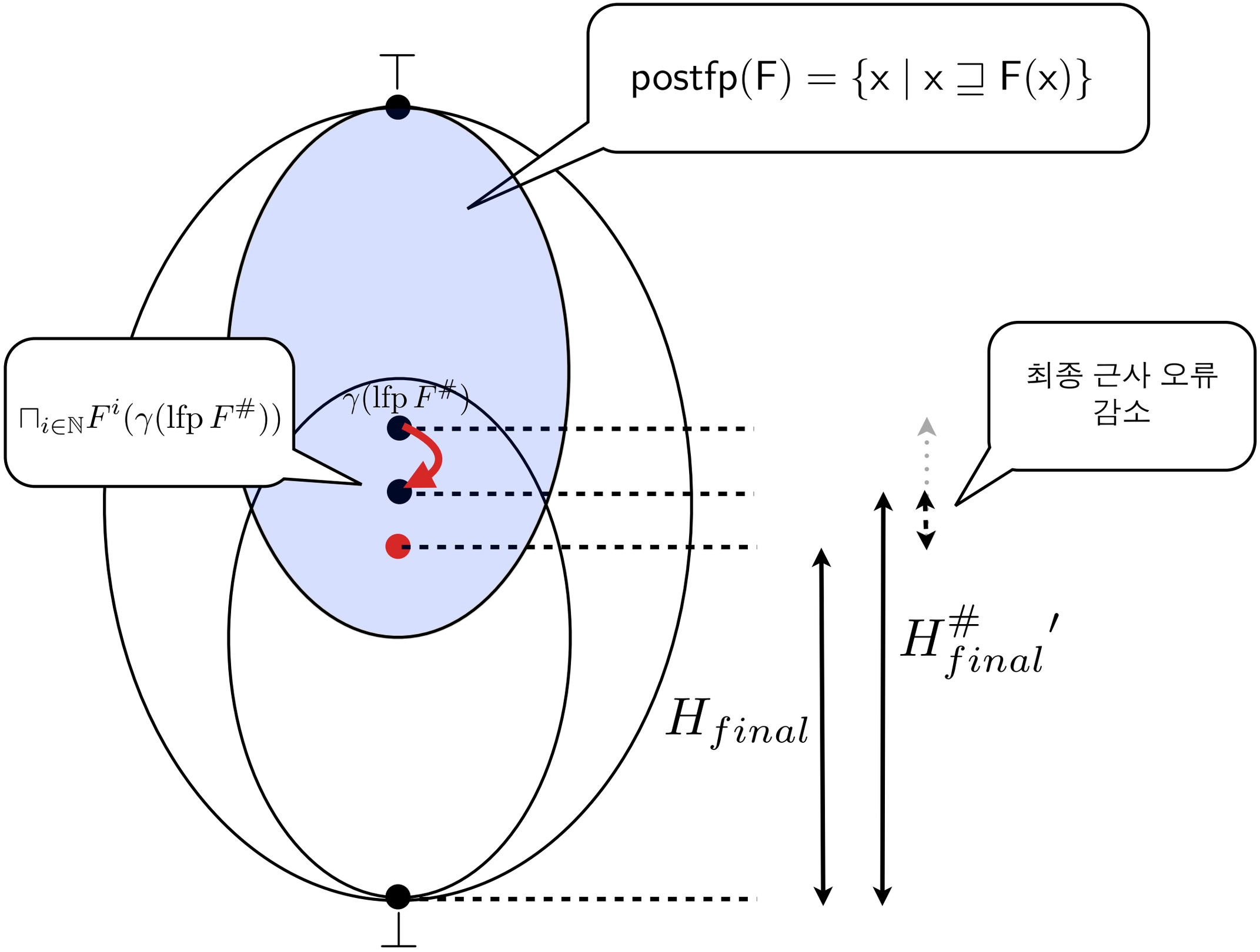
최종 래티스 높이 근사


사전분석
결과
최종분석
결과
$\gamma(\mathrm{lfp}\, F^{\#})$
$\mathrm{lfp}\, F$
최종 높이
근사 오류
H_{final}
$H^{\#}_{final}$

최종 래티스 높이 근사


$\mathsf{postfp}(\mathsf{F}) = \{\mathsf{x} \mid \mathsf{x} \sqsupseteq \mathsf{F}(\mathsf{x})\}$
$\top$
H_{final}
$H^{\#}_{final}$

최종 래티스 높이 근사


⊤
postfp(F) = {x | x ⊒ F(x)}
$\sqcap_{i\in\mathbb{N}} F^i(\gamma(\mathrm{lfp}\, F^{\#}))$
$\gamma(\mathrm{lfp}\, F^{\#})$
최종 근사 오류
감소
H_{final}
$H^{\#}_{final}{}'$

최종 래티스 높이 근사

- 하지만 사전분석 결과를 정확하게 만드는 것은 실제 분석을 수행하는 것과 다를바 없으므로 비용이 비쌈

- 대안 : 다음 정보를 이용하여 기계학습
  - 사전분석결과에 요약의미함수를 한두번 적용해봐서 얼마나 정확해지는지
  - 전분석결과
  - 프로그램 구조 정보

최종 래티스 높이 근사

Table 1. The feature vector used by linear regression to construct prediction models

Category	Feature
Inter-procedural (syntactic)	# function calls in the program
	# functions in recursive call cycles
	# undefined library function calls
Loop-related (syntactic)	the maximum loop size
	the average loop sizes
	the standard deviation of loop sizes
	the standard deviation of depths of loops
	# loopheads
Numerical analysis (semantic)	# bounded intervals in the pre-analysis result
	# unbounded intervals in the pre-analysis result
Pointer analysis (semantic)	# points-to sets of cardinality over 4 in the pre-analysis result
	# points-to sets of cardinality under 4 in the pre-analysis result
Post-fixpoint (semantic)	# program points where applying the transfer function once improves the precision
	height decrease when transfer function is applied once

실험결과

- 프로그래스 바의 질을 다음과 같이 수치화

  (n과 i는 각각 총 iteration, 현재 iteration 수, Best : 1)

$$Linearity(\bar{P}_i^{\#}) = 1 - \frac{\sum_{1 \le i \le n} (\frac{i}{n} - \bar{P}_i^{\sharp})^2}{\sum_{1 \le i \le n} (\frac{i}{n} - \frac{n+1}{2n})^2} \quad \in [0, 1]$$

- 8 GNU 프로그램에 대해서 테스트

인터벌 분석

Program	LOC	Time(s)		Linearity	Overhead	Height-Approx.
		Main	Pre			
bison-1.875	38841	3.66	0.91	0.73	24.86%	1.03
screen-4.0.2	44745	40.04	2.37	0.86	5.92%	0.96
lighttpd-1.4.25	56518	27.30	1.21	0.89	4.43%	0.92
a2ps-4.14	64590	32.05	11.26	0.51	35.13%	1.06
gnu-cobol-1.1	67404	413.54	99.33	0.54	24.02%	0.91
gnugo	87575	1541.35	7.35	0.89	0.48%	1.12
bash-2.05	102406	16.55	2.26	0.80	13.66%	0.93
sendmail-8.14.6	136146	1348.97	5.81	0.69	0.43%	0.93
TOTAL	686380	3423.46	130.5	**0.74**	**3.81%**	**Err : 0.07**

포인터 분석

Program	LOC	Time(s)		Linearity	Overhead	Height-Approx.
		Main	Pre			
screen-4.0.2	44745	15.89	1.56	0.90	9.82%	0.98
lighttpd	56518	11.54	0.87	0.76	7.54%	1.03
a2ps-4.14	64590	10.06	3.48	0.65	34.59%	1.04
gnu-cobol-1.1	67404	32.27	12.22	0.91	37.87%	1.03
gnugo	87575	217.77	3.88	0.64	1.78%	0.97
bash-2.05	102406	3.68	0.78	0.56	21.20%	1.04
proftpd-1.3.2	126996	74.64	11.14	0.82	14.92%	1.03
sendmail-8.14.6	136146	145.62	3.15	0.58	2.16%	0.98
TOTAL	686380	511.47	37.08	**0.73**	**7.25%**	**Err : 0.03**

옥타곤 분석

Program	LOC	Time(s)			Overhead
		Main	Pre	Linearity	
httptunnel-3.3	6174	49.5	8.2	0.91	16.6%
combine-0.3.3	11472	478.2	16	0.89	3.4%
bc-1.06	14288	63.9	43.8	0.96	68.6%
tar-1.17	18336	977.0	73.1	0.82	7.5%
parser	18923	190.1	104.8	0.97	55.1%
wget-1.9	35018	3895.36	1823.15	0.92	46.8%
TOTAL	69193	5654.0	2069.49	**0.91**	**36.6%**

추가비용으로 더 정확하게

- 전분석에서, 구분할 프로그램 지점을 더 세분화 하면 좋은결과를 얻을 수 있음

Program	**Linearity change**	**Overhead change**
bash-2.05 (pointer)	0.56 → 0.70	21.2% → 37.5%
sendmail-8.14.6 (interval)	0.69 → 0.95	0.4% → 18.4%

결론

- 분석기 프로그래스 바를 만들 수 있음
- 설계
  - 실제 분석보다 부정확, 그러나 빠른 전분석 설계
  - 정확할수록 작고 부정확할 수록 큰 수치값 부여하는 법 결정
- 진행
  - 전분석 후 본분석 진행
  - 전분석 수치값 및 수치값이 변화하는 양상이용, 본분석 프로그래스 바 구현

추천 기계학습 라이브러리 : scikit-learn

- http://scikit-learn.org/stable/