수정본이 여러 개라 원고의 대체 관계를 추가했습니다

지피터스 24기 · 자료 정리 2026-10-14 / 기존 실습 근거 2026-10-13 / 단독 재현 코드는 10-14 실행 확인

원고 버전을 고르는 기준

회차(Episode), 원고(Draft), 실행 근거(Evidence)를 연결한 전체 모델은 개체 11개·관계 12개입니다. 같은 회차에 연결된 원고 후보에서 supersedes로 대체된 항목을 제외하고, 하나만 남을 때 최신 원고를 선택합니다.

확인	실제 결과
E02 조회	E02-result-v2: 실제 실행 후 결과 원고
E01 조회	E01-revised: 10월 7일 수정본
최신 후보가 둘 또는 없음	확정하지 않고 ValueError

아래 단독 재현 코드는 전체 원본 파일을 요구하지 않도록 E02의 핵심 개체 4개·관계 5개만 사용합니다. 원본의 파일 존재 검사·자연어 검색은 이 축약 재현에 포함하지 않습니다. 전체 관계 JSON과 원작업 조회 코드는 뒤에 함께 실었습니다.

직접 확인

Python 3에서 `python3 knowledge-replay.py`를 실행합니다. PDF의 코드 부분을 같은 파일명으로 저장하고, 지면상 줄바꿈은 원래 코드의 줄과 구분해주세요. 출력 id가 E02-result-v2인지 확인합니다.

10월 14일 단독 재현 출력


```
{
  "id": "E02-result-v2",
  "type": "Draft",
  "version_label": "10/13 실제 실행 후 결과 원고",
  "source": "../reinforcement-20261013/team/harness-v2.md"
}
```

단독 재현 코드: knowledge-replay.py

긴 줄은 지면에 맞춰 자동 줄바꿈됩니다. 코드 복사 시 줄바꿈·들여쓰기를 확인하세요.

```
"""E02 latest-draft relationship replay; no original source files are opened."""
import json
GRAPH = json.loads(r'''{
  "entities": [
    {
      "id": "E02",
      "type": "Episode",
      "title": "AI 더치페이 도구를 믿어도 될까?",
      "next_action": "브라우저 계산 결과를 결과 원고에 반영",
      "source": "../life/episodes-current.json"
    },
    {
      "id": "E02-baseline",
      "type": "Draft",
      "source": "source/baseline-02.md",
      "version_label": "10/7 초안"
    },
    {
      "id": "E02-revised",
      "type": "Draft",
      "source": "source/revised-02.md",
      "version_label": "10/7 수정본"
    },
    {
      "id": "E02-result-v2",
      "type": "Draft",
      "version_label": "10/13 실제 실행 후 결과 원고",
      "source": "../reinforcement-20261013/team/harness-v2.md"
    }
  ],
  "relations": [
    {
      "from": "E02",
      "relation": "has_draft",
      "to": "E02-baseline"
    },
    {
      "from": "E02",
      "relation": "has_draft",
      "to": "E02-revised"
    },
    {
      "from": "E02",
      "relation": "has_draft",
      "to": "E02-result-v2"
    },
    {
      "from": "E02-result-v2",
      "relation": "supersedes",
      "to": "E02-revised"
```

```
    },
    {
      "from": "E02-revised",
      "relation": "supersedes",
      "to": "E02-baseline"
    }
  ]
}"")
def latest(graph, episode_id):
    entities = {x['id']: x for x in graph['entities']}
    candidates = {r['to'] for r in graph['relations']
                  if r['from'] == episode_id and r['relation'] == 'has_draft'}
    replaced = {r['to'] for r in graph['relations']
                if r['relation'] == 'supersedes' and r['from'] in candidates}
    survivors = candidates - replaced
    if len(survivors) != 1:
        raise ValueError('Latest draft is ambiguous or missing')
    return entities[next(iter(survivors))]
result = latest(GRAPH, 'E02')
assert result['id'] == 'E02-result-v2'
print(json.dumps(result, ensure_ascii=False, indent=2))
```

원작업 전체 모델: ontology.json


```
{
  "schema_version": 1,
  "types": [
    "Episode",
    "Draft",
    "Evidence"
  ],
  "relations_allowed": {
    "has_draft": [
      "Episode",
      "Draft"
    ],
    "supported_by": [
      "Episode",
      "Evidence"
    ],
    "supersedes": [
      "Draft",
      "Draft"
    ]
  },
  "entities": [
    {
      "id": "E01",
      "type": "Episode",
      "title": "AI 여행 일정은 실제로 움직일 수 있을까?",
      "next_action": "출발지와 여행 날짜 확인",
      "source": "../life/episodes-current.json"
    },
    {
      "id": "E02",
      "type": "Episode",
      "title": "AI 더치페이 도구를 믿어도 될까?",
      "next_action": "브라우저 계산 결과를 결과 원고에 반영",
      "source": "../life/episodes-current.json"
    },
    {
      "id": "E03",
      "type": "Episode",
      "title": "AI 실습 안내만으로 페이지를 만들 수 있을까?",
      "next_action": "사용 운영체제와 시험 참가자 확인",
      "source": "../life/episodes-current.json"
    },
    {
      "id": "E01-baseline",
      "type": "Draft",
      "source": "source/baseline-01.md",
      "version_label": "10/7 초안"
    },
    {
      "id": "E01-revised",
```

```
    "type": "Draft",
    "source": "source/revised-01.md",
    "version_label": "10/7 수정본"
   },
   {
    "id": "E02-baseline",
    "type": "Draft",
    "source": "source/baseline-02.md",
    "version_label": "10/7 초안"
   },
   {
    "id": "E02-revised",
    "type": "Draft",
    "source": "source/revised-02.md",
    "version_label": "10/7 수정본"
   },
   {
    "id": "E03-baseline",
    "type": "Draft",
    "source": "source/baseline-03.md",
    "version_label": "10/7 초안"
   },
   {
    "id": "E03-revised",
    "type": "Draft",
    "source": "source/revised-03.md",
    "version_label": "10/7 수정본"
   },
   {
    "id": "EV-SPLIT",
    "type": "Evidence",
    "source": "../evidence/split-tests.json"
   },
   {
    "id": "E02-result-v2",
    "type": "Draft",
    "version_label": "10/13 실제 실행 후 결과 원고",
    "source": "../reinforcement-20261013/team/harness-v2.md"
   }
  ],
  "relations": [
   {
    "from": "E01",
    "relation": "has_draft",
    "to": "E01-baseline"
   },
   {
    "from": "E01",
    "relation": "has_draft",
    "to": "E01-revised"
   },
   {
    "from": "E02",
    "relation": "has_draft",
    "to": "E02-baseline"
```

```
    },
    {
      "from": "E02",
      "relation": "has_draft",
      "to": "E02-revised"
    },
    {
      "from": "E03",
      "relation": "has_draft",
      "to": "E03-baseline"
    },
    {
      "from": "E03",
      "relation": "has_draft",
      "to": "E03-revised"
    },
    {
      "from": "E02",
      "relation": "supported_by",
      "to": "EV-SPLIT"
    },
    {
      "from": "E02",
      "relation": "has_draft",
      "to": "E02-result-v2"
    },
    {
      "from": "E02-result-v2",
      "relation": "supersedes",
      "to": "E02-revised"
    },
    {
      "from": "E01-revised",
      "relation": "supersedes",
      "to": "E01-baseline"
    },
    {
      "from": "E02-revised",
      "relation": "supersedes",
      "to": "E02-baseline"
    },
    {
      "from": "E03-revised",
      "relation": "supersedes",
      "to": "E03-baseline"
    }
  ]
}
```

원작업 조회 코드: query.py (참고용)

원작업의 전체 원고·근거 파일이 필요한 코드입니다. 이 부록만 복사해 단독 실행할 수는 없습니다. 독립 실행은 앞의 knowledge-replay.py를 사용하세요. PDF에서 코드를 복사하면 문자열 중간에 지면 줄바꿈이 들어갈 수 있어 원래 줄을 확인해야 합니다.

```
"""Resolve the latest recorded draft by explicit supersedes edges, not a filename guess."""
import argparse,json
from pathlib import Path
p=Path(__file__).parent
def query(episode_id):
 g=json.loads((p/'ontology.json').read_text());entities={x['id']:x for x in g['entities']}
 if len(entities)!=len(g['entities']):raise ValueError('Duplicate entity ID')
 for r in g['relations']:
  assert r['from'] in entities and r['to'] in entities
  assert [entities[r['from']]['type'],entities[r['to']]['type']]==g['relations_allowed'][r['relation']]
 for e in entities.values():assert (p/e['source']).resolve().exists(),e['source']
 if episode_id not in entities or entities[episode_id]['type']!='Episode':raise ValueError('Unknown episode')
 draft_ids={r['to'] for r in g['relations'] if r['relation']=='has_draft' and r['from']==episode_id}
 replaced={r['to'] for r in g['relations'] if r['relation']=='supersedes' and r['from'] in draft_ids}
 latest=draft_ids-replaced
 if len(latest)!=1:raise ValueError('Latest draft ambiguous or cycle: '+str(sorted(latest)))
 winner=entities[next(iter(latest))]
 return {'entities':len(entities),'relations':len(g['relations']),'episode':episode_id,'latest_draft':winner,'drafts':[entities[i] for i
in sorted(draft_ids)],'evidence':[entities[r['to']] for r in g['relations'] if r['relation']=='supported_by' and
r['from']==episode_id],'rule':'same-episode has_draft candidates minus drafts replaced by supersedes; exactly one
survivor required'}
if __name__=='__main__':
 parser=argparse.ArgumentParser();parser.add_argument('--
episode',default='E02');args=parser.parse_args();print(json.dumps(query(args.episode),ensure_ascii=False,indent=2))
```