1. 과제 선정 내용

✓ 인사/복리후생 규정 중, 자사 경조사비용 지급 기준

인사/복리후생 규정

채용/복무 규정
포상/징계 규정
근태/퇴직 규정
휴일/휴가 실시 기준
급여 규정
인사평가 규정
인사 체계
장학 제도
복리 제도
경조 제도
직원 대여금 제도

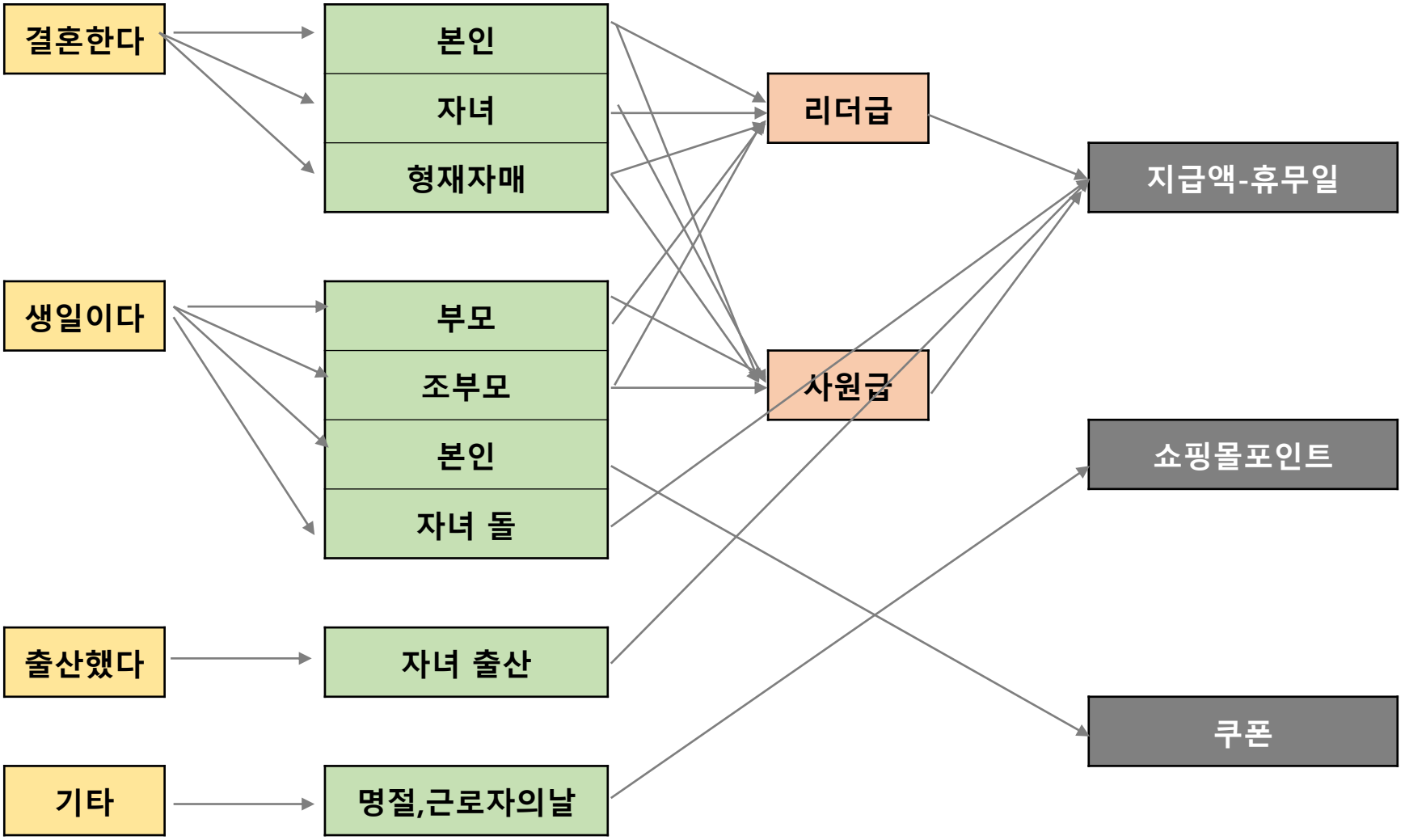
■ 경조사 지급 기준

(리더급:과장이상 / 사원급:대리,사원) 단위:원

구분	해당사항	임원 (지급율)	리더급	사원급	휴일수	비고
결혼	본인	50%	1,500,000	1,000,000	6	화환(항공료)
	자녀	25%	750,000	500,000	2	
	(처의)형제자매	10%	300,000	200,000	1	
환갑/칠순	(처의)부모	10%	300,000	200,000	1	
	(외)조부모	5%	150,000	100,000	1	
사망 (과제 제외)	본인	100%	3,000,000	2,000,000		조화
	배우자	50%	1,500,000	1,000,000	6	조화(항공료)
	(처의)부모	30%	900,000	600,000	6	조화(항공료)
	자녀	30%	900,000	600,000	3	조화(항공료)
	형제자매	10%	300,000	200,000	3	
	조부모	20%	400,000	300,000	3	
	외조부모	20%	400,000	300,000	3	
	백숙부모				2	
	처의 형제자매				1	
	탈상(부모,승중상)				1	
기타	자녀출산				3	꽃바구니
	자녀 돌 기념품	200,000				
	설날/추석/ 근로자의 날	선물(10 만원 상당)				
	본인 생일	생일케이크(2 만원 상당)				

2. 과제 추론 방법

✓ 경조사비용 지급 기준 추론



Open in Colab

In [1]:
```
!pip install durable_rules
from durable.lang import *
```

```
Requirement already satisfied: durable_rules in /usr/local/lib/python3.7/dist-packages (2.0.28)
```

In [2]:
```
psum = 0
dsum = 0

def paySum(pay):
  psum += pay

def daySum(day):
  dsum += day
```

In [3]:
```
with ruleset('employee'):
    @when_all(c.first << (m.predicate == '나는') & (m.object == '결혼한다'),
              (m.predicate == '나는') & (m.object == '리더급이다') & (m.subject == c.first.subject))
    def A(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼축하금', 'object': 'pay150' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼휴일일수', 'object': 'day6' })

    @when_all(c.first << (m.predicate == '나는') & (m.object == '결혼한다'),
              (m.predicate == '나는') & (m.object == '사원급이다') & (m.subject == c.first.subject))
    def B(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼축하금', 'object': 'pay100' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼휴일일수', 'object': 'day6' })

    @when_all(c.first << (m.predicate == '자녀가') & (m.object == '결혼한다'),
              (m.predicate == '나는') & (m.object == '리더급이다') & (m.subject == c.first.subject))
    def C(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼축하금', 'object': 'pay75' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼휴일일수', 'object': 'day2' })

    @when_all(c.first << (m.predicate == '형제자매(배우자포함)') & (m.object == '결혼한다'),
              (m.predicate == '나는') & (m.object == '사원급이다') & (m.subject == c.first.subject))
    def D(c):
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼축하금', 'object': 'pay50' })
        c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼휴일일수', 'object': 'day2' })
```

```
@when_all(c.first << (m.predicate == '형제자매(배우자포함)') & (m.object == '결혼한다'),
          (m.predicate == '나는') & (m.object == '리더급이다') & (m.subject == c.first.subject))
def E(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼축하금', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '자녀가') & (m.object == '결혼한다'),
          (m.predicate == '나는') & (m.object == '사원급이다') & (m.subject == c.first.subject))
def F(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼축하금', 'object': 'pay20' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '결혼휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '부모(처포함)') & (m.object == '회갑/칠순이다'),
          (m.predicate == '나는') & (m.object == '리더급이다') & (m.subject == c.first.subject))
def G(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순경조금', 'object': 'pay30' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '부모(처포함)') & (m.object == '회갑/칠순이다'),
          (m.predicate == '나는') & (m.object == '사원급이다') & (m.subject == c.first.subject))
def H(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순경조금', 'object': 'pay20' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '(외)조부모') & (m.object == '회갑/칠순이다'),
          (m.predicate == '나는') & (m.object == '리더급이다') & (m.subject == c.first.subject))
def I(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순경조금', 'object': 'pay10' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '(외)조부모') & (m.object == '회갑/칠순이다'),
          (m.predicate == '나는') & (m.object == '사원급이다') & (m.subject == c.first.subject))
def J(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순경조금', 'object': 'pay10' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '회갑/칠순휴일일수', 'object': 'day1' })

@when_all(c.first << (m.predicate == '자녀를') & (m.object == '출산한다'))
def K(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '출산휴가', 'object': 'day10' })

@when_all(c.first << (m.predicate == '자녀가') & (m.object == '둘이다'))
def L(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '자녀둘', 'object': 'pay20' })

@when_any(c.first << (m.predicate == '나는') & ((m.object == '리더급이다') | (m.object == '사원급이다')))
def M(c):
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '명절/근로자의날', 'object': 'point' })
    c.assert_fact({ 'subject': c.first.subject, 'predicate': '본인생일', 'object': 'coupon' })
```

```
@when_any((m.object == 'pay150') & ((m.predicate == '결혼축하금')))
def pay150(c):
    print('Fact: {0}에게 {1}은 150만원을 지급한다{2}'.format(c.m.subject, c.m.predicate, psum))
    psum += 150
    paySum(150)

@when_any((m.object == 'pay100') & ((m.predicate == '결혼축하금') | (m.predicate == '회갑/칠순경조금')))
def pay100(c):
    print('Fact: {0}에게 {1}은 100만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 100
    paySum(100)

@when_any((m.object == 'pay75') & ((m.predicate == '결혼축하금') | (m.predicate == '회갑/칠순경조금')))
def pay75(c):
    print('Fact: {0}에게 {1}은 75만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 75

@when_any((m.object == 'pay50') & ((m.predicate == '결혼축하금') | (m.predicate == '회갑/칠순경조금')))
def pay50(c):
    print('Fact: {0}에게 {1}은 50만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 50

@when_any((m.object == 'pay30') & ((m.predicate == '결혼축하금') | (m.predicate == '회갑/칠순경조금')))
def pay30(c):
    print('Fact: {0}에게 {1}은 30만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 30

@when_any((m.object == 'pay20') & ((m.predicate == '결혼축하금') | (m.predicate == '회갑/칠순경조금') | (m.predicate == '자녀돌')))
def pay20(c):
    print('Fact: {0}에게 {1}은 20만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 20

@when_any((m.object == 'pay15') & ((m.predicate == '결혼축하금') | (m.predicate == '회갑/칠순경조금') | (m.predicate == '자녀돌')))
def pay15(c):
    print('Fact: {0}에게 {1}은 15만원을 지급한다'.format(c.m.subject, c.m.predicate))
    psum += 15

@when_any((m.object == 'day10'))
def day6(c):
    print('Fact: {0}에게 {1}는 10일 및 꽃바구니 지급합니다'.format(c.m.subject, c.m.predicate))
    dsum += 10

@when_any((m.object == 'day6'))
def day6(c):
    print('Fact: {0}에게 {1}는 6일 입니다'.format(c.m.subject, c.m.predicate))
    dsum += 6

@when_any((m.object == 'day2'))
def day2(c):
    print('Fact: {0}에게 {1}는 2일 입니다'.format(c.m.subject, c.m.predicate))
    daySum(2)
```

```
    @when_any((m.object == 'day1'))
    def day1(c):
        print('Fact: {0}에게 {1}는 1일 입니다'.format(c.m.subject, c.m.predicate))
        dsum += 1

    @when_any((m.object == 'point'))
    def point(c):
        print('Fact: {0}에게 {1}에는 자사쇼핑몰에서 각각 10만원(총 30만원) 포인트를 지급한다'.format(c.m.subject, c.m.predicate))
        psum += 30

    @when_any((m.object == 'coupon'))
    def coupon(c):
        print('Fact: {0}에게 {1}에는 카카오 케익쿠폰(2만원)을 지급한다'.format(c.m.subject, c.m.predicate))
        psum += 2

    #@when_all(+m.subject)
    #def output(c):
    #    print('assert_fact 등록내용: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.object))
```

In [4]:
```
assert_fact('employee', { 'subject': '김길동', 'predicate': '나는', 'object': '결혼한다' })
assert_fact('employee', { 'subject': '김길동', 'predicate': '나는', 'object': '리더급이다' })
assert_fact('employee', { 'subject': '김길동', 'predicate': '부모(처포함)', 'object': '회갑/칠순이다' })
assert_fact('employee', { 'subject': '김길동', 'predicate': '자녀가', 'object': '돌이다' })
assert_fact('employee', { 'subject': '김길동', 'predicate': '자녀를', 'object': '출산한다' })
assert_fact('employee', { 'subject': '김길동', 'predicate': '형제자매(배우자포함)', 'object': '결혼한다' })

print("올해지급 받는 총액은 {0}만원이며, 연차외의 총 경조휴무일은 {1}일 입니다".format(psum, dsum))
```

```
Fact: 김길동에게 결혼휴일일수는 6일 입니다
Fact: 김길동에게 명절/근로자의날에는 자사쇼핑몰에서 각각 10만원(총 30만원) 포인트를 지급한다
Fact: 김길동에게 본인생일에는 카카오 케익쿠폰(2만원)을 지급한다
Fact: 김길동에게 회갑/칠순경조금은 30만원을 지급한다
Fact: 김길동에게 회갑/칠순휴일일수는 1일 입니다
Fact: 김길동에게 자녀돌은 20만원을 지급한다
Fact: 김길동에게 출산휴가는 10일 및 꽃바구니 지급합니다
Fact: 김길동에게 결혼축하금은 30만원을 지급한다
Fact: 김길동에게 결혼휴일일수는 1일 입니다
올해지급 받는 총액은 0만원이며, 연차외의 총 경조휴무일은 0일 입니다
```

In [4]: