Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems

Hyunjong Choi, Hyoseung Kim, Qi Zhu†
Outline

I Introduction
II Related Work & Motivation
III Job-class-level Scheduling
IV Schedulability Analysis
V Evaluation
VI Conclusion and Future Work
Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work
Weakly-hard real-time systems

- Many practical systems
  - Tolerable to some deadline misses w/o affecting functional correctness

\( (m, K) \): at most \( m \) jobs can miss their deadlines among any \( K \) consecutive jobs

Effectiveness of weakly-hard real-time systems

- Navigation of an autonomous vehicle in Gazebo with ROS
  - A periodic task: ControlTask
  - Mission: Drive from start to end points
  - Injected deadline misses w.r.t. weakly-hard constraints

Tasks with bounded deadline misses can produce a functional correctness

Resource can be reserved for the other tasks

† It sends velocity command to robot base(actuator) at the specified rate defined as a control frequency.
Limitation of task-level fixed-priority scheduling

- Simple taskset with weakly-hard constraints

![Diagram showing task execution with deadlines]

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 = 11, C_1 = 6, m_1 = 2, K_1 = 4 )</td>
<td>( T_2 = 7, C_2 = 4, m_2 = 4, K_2 = 7 )</td>
</tr>
</tbody>
</table>

**Specifications**

No matter which task has a higher priority, NOT schedulable!

New approach

\[ \text{\dag} \text{Task experiences more than } m \text{ deadline misses in a window of } K \text{ jobs.} \]
Contributions

- Main contributions
  - Propose a new job-class-level fixed-priority scheduler based on meet-oriented classification of jobs of tasks
  - Present the schedulability analysis framework for our proposed scheduler
  - Generalization of task-level fixed-priority scheduling
  - Outperforms the latest work in terms of task schedulability, analysis running time
  - Implement our scheduler in the Linux kernel running on Raspberry Pi
Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work
System Model

- Task model
  - \( \tau_i := (C_i, D_i, T_i, (m_i, K_i)) \)
    - \( C_i \): The worse-case execution time
    - \( D_i \): The relative deadline
    - \( T_i \): The minimum inter-arrival time
    - \( (m_i, K_i) \): The weakly-hard constraints \( m_i < K_i \). For a hard real-time task, \( m_i = 0 \) and \( K_i = 1 \).

- Preemptive scheduling

- Uniprocessor system
Job-Class-Level Fixed-Priority Scheduling

- Job classification
  - Assign different priorities to individual job-classes

  *Meet-oriented : the number of prior deadlines consecutively met*

- For instance, \((m, K) = (2, 4)\) can have job classes: \(J^0\), \(J^1\), and \(J^2\)
- Priority of a job-class decrease monotonically
Bounding consecutive deadline misses

- Miss threshold $w_i$
  - Limit the distance from the current job to the previous deadline-met jobs to bound the number of consecutive deadline misses

$$w_i = \max \left( \left\lfloor \frac{K_i}{K_i - m_i} \right\rfloor - 1, 1 \right)$$

- Ensure enough number of jobs running with the highest priority job-class
- For instance, $(m, K) = (5, 7)$ where $w_i = 2$ allows 2 consecutive deadline misses
Priority assignment

- A heuristic priority assignment
  - An extension of the deadline monotonic (DM) priority assignment

**Rule.**

- Assign higher priority to a job-class with a smaller index
- For job-classes with the same index,
  - Higher priority to shorter deadline ($q = 0$)
  - Higher priority to shorter miss threshold with deadlines for tie-breaking ($q > 0$)

Subsumes the task-level DM priority assignment

---

**Lemma 3.**

**Algorithm 1** Job-class priority assignment

<table>
<thead>
<tr>
<th>Input: $\Gamma$: Taskset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $N \leftarrow</td>
</tr>
<tr>
<td>2: Sort $\tau_i$ in $\Gamma$ in ascending order of deadline</td>
</tr>
<tr>
<td>3: for all $\tau_i \in \Gamma$ do</td>
</tr>
<tr>
<td>4: $l_i \leftarrow K_i - m_i + 1$ \textgreater; $l_i$: number of job-classes for $\tau_i$</td>
</tr>
<tr>
<td>5: end for</td>
</tr>
<tr>
<td>6: $\text{prio} \leftarrow \sum_{i=1}^{N} l_i$ \textgreater; Priority to be assigned next</td>
</tr>
<tr>
<td>7: if $\Gamma$ is schedulable by DM then</td>
</tr>
<tr>
<td>8: for all $\tau_i \in \Gamma$ do</td>
</tr>
<tr>
<td>9: \textgreater; Assign the same priority to all job-classes of $\tau_i$</td>
</tr>
<tr>
<td>10: for all $q \leftarrow 0$ to $l_i - 1$ do</td>
</tr>
<tr>
<td>11: $\pi_i^q \leftarrow \text{prio}$</td>
</tr>
<tr>
<td>12: end for</td>
</tr>
<tr>
<td>13: $\text{prio} \leftarrow \text{prio} - 1$</td>
</tr>
<tr>
<td>14: end for</td>
</tr>
<tr>
<td>15: else</td>
</tr>
<tr>
<td>16: $L \leftarrow \max_{\tau_i \in \Gamma} l_i$</td>
</tr>
<tr>
<td>17: for $q \leftarrow 0$ to $L - 1$ do</td>
</tr>
<tr>
<td>18: if $q &gt; 0$ then</td>
</tr>
<tr>
<td>19: Sort $\tau_i$ in $\Gamma$ in ascending order of $w_i$ and deadline</td>
</tr>
<tr>
<td>20: end if</td>
</tr>
<tr>
<td>21: for all $\tau_i \in \Gamma$ do</td>
</tr>
<tr>
<td>22: if $q &lt; l_i$ then \textgreater; Check if $q$ is a valid index</td>
</tr>
<tr>
<td>23: $\pi_i^q \leftarrow \text{prio}$</td>
</tr>
<tr>
<td>24: $\text{prio} \leftarrow \text{prio} - 1$</td>
</tr>
<tr>
<td>25: end if</td>
</tr>
<tr>
<td>26: end for</td>
</tr>
<tr>
<td>27: end for</td>
</tr>
<tr>
<td>28: end if</td>
</tr>
</tbody>
</table>
An example of job-class-level scheduling

- With the same taskset at page 6.

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
</tr>
<tr>
<td>Task 2</td>
</tr>
</tbody>
</table>

Schedulable !!
Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work
Schedulability Analysis

- The schedulability analysis of tasks with weakly-hard constraints under job-class-level scheduling decompose

Step 1: Analyzing the WCRT of each job-class
Extension of WCRT in task-level

Step 2: Finding all possible job-class patterns
Used reachability tree

< Schedulability analysis process of job-class-level scheduler >
Worse-case response time of job-classes

- Worse-case response time of $J_i^q$ is bounded by the recurrence:

$$R_i^{q,n+1} \leftarrow C_i + \sum_{\tau_k \in \Gamma - \tau_i} W_i^q (R_i^{q,n}, \tau_k)$$

$W_i^q$ is an upper-bound of interference imposed on $J_i^q$

$$W_i^q(t, \tau_k) = \min \left( \sum_{\forall p : \pi_i^q < \pi_k^p} \left\lfloor \frac{t + J_k}{\eta(J_k^p)} \right\rfloor \times C_k, \left\lfloor \frac{t + J_k}{T_k} \right\rfloor \times C_k \right)$$

- Each job-class has a different minimum job-class inter-arrival time, $\eta(J_k^p)$

Lemma 8.

Generalization of the task-level iterative response time test for hard real-time tasks.

Schedulability check

- Schedulability test of a task with \( m_i/K_i \geq 0.5 \)

Lemma 10.

A task \( \tau_i \) is always schedulable if the ratio of \( m_i/K_i \) is greater than or equal to 0.5 and it satisfies the prerequisite given by Lemma 9.

Step 1: Show at least 1 deadline met in \( K_i \) window by using a necessary condition

\[
(w_i + 1) \cdot \alpha \leq K_i
\]

\[
\text{WCRT}(J_i^0) \leq D_i
\]

Step 2: Show that the number of deadline met satisfies the constraint

\[
\frac{1}{w_i+1} \geq \frac{K_i-m_i}{K_i} \quad \Rightarrow \quad \left| \frac{K_i}{K_i-m_i} \right| \leq \frac{K_i}{K_i-m_i}
\]

Always true as \( m_i \leq K_i - 1 \)
Reachability tree

- For tasks with $m_i/K_i < 0.5$, find all possible job-class patterns for $K_i$ job executions using *reachability tree*

**Lemma 13.**

The reachability trees of a task $\tau_i$ represent all possible job-class patterns that the task can experience at its runtime for $K_i$ execution window
Implementation cost

- Measure runtime overhead of the proposed scheduler implementation

- Experimental setup
  - Linux kernel v4.9.80 running on Raspberry Pi 3
  - ARM Cortex-A53 @ clock frequency of 1.2 GHz
  - Run 5 tasks with period of 20ms to 40ms for 10 minutes (118,569 jobs)

<table>
<thead>
<tr>
<th>Type</th>
<th>Mean</th>
<th>Max</th>
<th>Min</th>
<th>99%th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updating (\mu)-pattern†</td>
<td>0.3002</td>
<td>1.1460</td>
<td>0.1040</td>
<td>0.6250</td>
</tr>
<tr>
<td>Updating job-class index</td>
<td>1.5035</td>
<td>11.8750</td>
<td>0.5210</td>
<td>2.5000</td>
</tr>
<tr>
<td>Changing task priority</td>
<td>4.7633</td>
<td>28.9580</td>
<td>3.0210</td>
<td>11.3020</td>
</tr>
<tr>
<td>Rollback</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checkpointing</td>
<td>1.9413</td>
<td>9.3230</td>
<td>1.2500</td>
<td>3.2290</td>
</tr>
<tr>
<td>Recovery</td>
<td>6.1257</td>
<td>24.8430</td>
<td>0.4680</td>
<td>8.3146</td>
</tr>
</tbody>
</table>

*< Runtime overhead [\(\mu s\)]>*

† Represents a sequence of deadline met and missed jobs of a task, (G. Bernat, A. Burns, and A. Liamosi. “Weakly hard real-time systems”, 2001)
Schedulability experiments

- The evaluation is conducted in two ways:
  - **Comparison** with other weakly-hard scheduling schemes (WSA†, RTO-RM*)
    - WSA: delayed completion for deadline-missed jobs
    - RTO-RM: job abort for deadline-missed jobs
  - **Exploration** of the proposed scheduler under diverse experimental conditions
  - Performance metric: percentage of schedulable taskset, analysis running time

- Taskset generation

<table>
<thead>
<tr>
<th>Number of tasksets</th>
<th>Task utilization (UUniFast algorithm#)</th>
<th>Task period [ms]</th>
<th>K range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1,000</td>
<td>[0.8, 1.8]</td>
<td>[10, 1000]</td>
</tr>
</tbody>
</table>

† Y. Sun and M. D. Natale, “Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks,” *TECS*, 2017
* G. Koren and D. Shasha. “Skip-over: Algorithms and complexity for overloaded systems that allow skips”, RTSS, 1995
Taskset schedulability

- Comparison of schedulability ratio with other schemes
  - 1,000 tasksets with 20 tasks
  - $K_i = 10, m_i = [1, 9]$, common $(m, K)$ for a taskset

JCLS better utilizes CPU resource when there are overloaded weakly-hard tasksets
Analysis running time

- Time to determine the schedulability of a given taskset
  - By the number of tasks in a taskset (10, 30, and 50 tasks)
  - 1,000 tasksets, $K_i = 10, m_i = [1, 9]$
  - JCLS (on Raspberry Pi 3), WSA (on Intel Core-i7 for CPLEX Optimizer)

<table>
<thead>
<tr>
<th>Number of tasks</th>
<th>Approach</th>
<th>Mean</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>JCLS</td>
<td>0.0010</td>
<td>0.0046</td>
</tr>
<tr>
<td></td>
<td>WSA</td>
<td>0.2739</td>
<td>114.2892</td>
</tr>
<tr>
<td>30</td>
<td>JCLS</td>
<td>0.0112</td>
<td>0.0432</td>
</tr>
<tr>
<td></td>
<td>WSA</td>
<td>25.7284</td>
<td>1800.5996</td>
</tr>
<tr>
<td>50</td>
<td>JCLS</td>
<td>0.0331</td>
<td>0.1463</td>
</tr>
<tr>
<td></td>
<td>WSA</td>
<td>78.5982</td>
<td>3002.5189</td>
</tr>
</tbody>
</table>

The analysis time of JCLS is shorter than that of WSA
More applicable to runtime admission control
Conclusion & Future work

- **Conclusion**
  - **New job-class-level fixed-priority scheduling** and **analysis** for weakly-hard real-time systems
  - Proposed scheduler **outperforms prior work** with respect to taskset schedulability and analytical complexity
  - Proposed approach is effective in overloaded situations (e.g., maximum utilization is higher than 1)

- **Future work**
  - Address the pessimism of our schedulability analysis when the ratio of $m_i/K_i$ is less than 0.5
Thank you

Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems

Hyunjong Choi, Hyoseung Kim, Qi Zhu

Q & A
Appendix

1. Related work
2. Utilizations
3. Benefits of the meet-oriented classification
4. Minimum time interval of a job-class
5. Interference of job-class-level analysis
6. Schedulability check
7. Complexity of reachability tree
8. An example of reachability tree
Related work

- Goals in weakly-hard systems: **guarantee & improve schedulability**
  - Scheduling: task-level fixed-priority scheduling
  - Assumptions: *initial offset is known*, *periodic task* with no jitter

  **Limits applicability to recent cyber physical systems**

- †, ‡ Bernat et al. works on the schedulability of periodic tasks with weakly-hard constraints under fixed-priority scheduling (RTSS’2001)
- Typical worst-case analysis (TWCA) approaches significantly contributes to weakly-hard systems (DATE’2012, DATE’2013, EMSOFT’2014, ECRTS’2015)
  - Assume exact arrival patterns of task instances is known
- ◆ Sun et al. relaxed the assumption on offset and jitter (TECS’2017)
- †, ◆, ‡ Goossens et al. distanced-based dynamic-priority scheduling (RTNS’2008)
Utilizations

- Represent the resource usage
  - Maximum utilization
    
    **Definition 1.**
    
    Maximum utilization of a task $\tau_i$, $U_i^M$, is the maximum amount of CPU resource that $\tau_i$ can utilize, defined as $U_i^M = \frac{C_i}{T_i}$

    *Maximum total utilization: $U^M = \sum_{i=1}^{N} C_i / T_i$

- Minimum utilization

  **Definition 2.**
  
  Minimum utilization of a task $\tau_i$, $U_i^m$, is the CPU resource used by $\tau_i$ when it experiences the maximum deadline misses allowed by its $(m_i, K_i)$ constraint, i.e., $U_i^m = \frac{C_i}{T_i} \times \frac{K_i - m_i}{K_i}$

  *Minimum total utilization: $U^m = \sum_{i=1}^{N} \frac{C_i}{T_i} \times \frac{K_i - m_i}{K_i}$
Benefits of the *meet-oriented* classification

- Benefits of meet-oriented classification
  - It reduces interferences imposed by higher priority jobs by modulating consecutive meets
  - Enables to avoid a pessimism when we evaluate WCRT of a job.

<table>
<thead>
<tr>
<th>Job-classes</th>
<th>J₀</th>
<th>J¹</th>
<th>J²</th>
<th>J³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meet/Miss</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>○</td>
</tr>
<tr>
<td>Priorities</td>
<td>Low</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

- May miss or meet
- Always meet

< After scheduling >

- Deadline missed
- Deadline met

< Consecutive execution of high-priority jobs under *miss-oriented* job classification >

\( m, K = (3, 6) \)

3 consecutive high priority jobs

3 missed jobs

\( J₉^{1,5} \) release

\( J₉^{1,6} \) release

\( J₉^{2,7} \) release

Start scheduling
Minimum job-class inter-arrival time (1/4)

- As a first step, analyzing the WCRT of individual job-classes
- Upper bound the maximum interference imposed by the jobs of other tasks with higher-priority job-classes

\[ q = K_i - m_i \]

- \[ \eta(J_i^q) = 1 \cdot T_i \]

\[ q < K_i - m_i \& \ WCRT(J_i^q) > D_i \]

- \[ \eta(J_i^q) = (q + 1) \cdot T_i, \text{if } w_i = 1 \]
- \[ \eta(J_i^q) = 1 \cdot T_i, \text{if } w_i > 1 \]

\[ q < K_i - m_i \& \ WCRT(J_i^q) \leq D_i \]

- \[ \eta(J_i^q) = (w_i + 1) \cdot T_i, \text{if } q = 0 \]
- \[ \eta(J_i^q) = (q + 2) \cdot T_i, \text{if } q > 0 \]

Ex) Maximum job-class index : 3

<table>
<thead>
<tr>
<th></th>
<th>J^0</th>
<th>J^1</th>
<th>J^2</th>
<th>J^3</th>
<th>J^?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miss</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) WCRT\((J_i^{K_i-m_i}) \leq D_i \) Deadline met

2) WCRT\((J_i^{K_i-m_i}) > D_i \) Deadline missed

Worst case, \( \eta(J_i^q) = 1 \cdot T_i \)
Minimum time interval of a job-class (2/4)

- A job-class whose the WCRT > $D_i$,

**Lemma 5.**

The minimum inter-arrival time of $J_i^q$ where $q < K_i - m_i$ and the WCRT of $J_i^q$ is greater than $D_i$ is given by

$$\eta(J_i^q) = \begin{cases} (q + 1) \cdot T_i, & \text{if } w_i = 1 \\ 1 \cdot T_i, & \text{if } w_i > 1 \end{cases}$$

---

**Diagram:**

- Deadline met
- Deadline missed

- $\eta(J_i^0) = T_i$
- $\eta(J_i^1) = 2 \cdot T_i$
- $\eta(J_i^2) = 3 \cdot T_i$

WCRT > $D_i$ and $w_i = 1$
Minimum time interval of a job-class (3/4)

- A job-class whose the WCRT > $D_i$,

Lemma 5.

The minimum inter-arrival time of $J_i^q$ where $q < K_i - m_i$ and the WCRT of $J_i^q$ is greater than $D_i$ is given by

$$\eta(J_i^q) = \begin{cases} 
(q + 1) \cdot T_i, & \text{if } w_i = 1 \\
1 \cdot T_i, & \text{if } w_i > 1
\end{cases}$$

WCRT > $D_i$ and $w_i > 1$
A job-class whose the WCRT $\leq D_i$,

**Lemma 6.**
The minimum inter-arrival time of $J_i^{q}$ where $q < K_i - m_i$ and the WCRT of $J_i^{q}$ is less than or equal to $D_i$ is given by

$$\eta(J_i^{q}) = \begin{cases} 
(w_i + 1) \cdot T_i, & \text{if } q = 0 \\
(q + 2) \cdot T_i, & \text{if } q > 0
\end{cases}$$
Interference of job-class-level analysis

- An upper-bound of interference imposed on $J_i^q$ by the higher priority jobs $J_k^p$ of another tasks during arbitrary time $t$
  - Extension of previous work†

\[ R_i = C_i + I_i, \quad I_i = \sum_{j=1}^{i-1} \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j \]

\[ W_i^q(t, \tau_k) = \min \left( \sum_{\forall p: \pi_i^q < \pi_k^p} \left[ \frac{t + J_k}{\eta(J_k^p)} \right] \times C_k \left[ \frac{t + J_k}{T_k} \right] \times C_k \right) \]

\[ \checkmark J_k \text{ is a jitter of a higher priority job} \]

Worse-case response time of job-classes

- Worse-case response time of $J_i^q$ is bounded by the recurrence:

\[
R_{i}^{q,n+1} \leftarrow C_i + \sum_{\tau_k \in \Gamma - \tau_i} W_i^q (R_{i}^{q,n}, \tau_k)
\]

Theorem 1.

✓ $\Gamma$ is the entire taskset
✓ Starts with $R_{i}^{q,0} = C_i$ and terminates when $R_{i}^{q,n} + J_i > D_i$ or $R_{i}^{q,n+1} = R_{i}^{q,n}$

Lemma 8.

The job-class-level response time test for weakly-hard tasks given in Theorem 1 is a generalization of the task-level iterative response time test for hard real-time tasks.
Schedulability check

Theorem 2.

A task is guaranteed to be schedulable if the $\mu$-patterns at all leaf nodes in its reachability trees satisfy the weakly-hard constraint.
Complexity of a reachability tree

- Inspecting all possible patterns is an inefficient way?
  - However, in a reachability tree, the upper-bound on the number of nodes follows the Fibonacci sequence \( f_{i+2} = f_{i+1} + f_i \)
  - For a task \( \tau_i \), the upper-bound of complexity of computing all the reachability trees is represented as

\[
O_i \leq (K_i - m_i + 1) \times \frac{\rho^{K_i + 1} - (1 - \rho)^{K_i + 1}}{\sqrt{5}}
\]

Where \( \rho = \frac{1+\sqrt{5}}{2} \) which is golden ratio and \( K_i - m_i + 1 \) is the number of job-classes

Theorem 4.†

An example of reachability tree

- A job-class $J^0_1$ of Task 1

$q$, $m$ : $\mu$–patterns

$q$ : index of a job-class
$m$ : number of misses

0 : miss
Miss
1 : meet
Meet

$C$–patterns

$\mu$–patterns

$C^q_m$ : number of misses

$q$ : index of a job-class

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns:

$\mu$–patterns

$J^q_m$ : $C$–patterns: